Alarm Gateway Object for Wonderware Application Server

User Guide
Ver 1.x Rev 1.10
PR 00185

WONDERWARE FINLAND
P.O. Box 38
FIN-00371 Helsinki Finland
tel. int. + 358 9 5404940
fax int. + 358 9 5413541
www.wonderware.fi

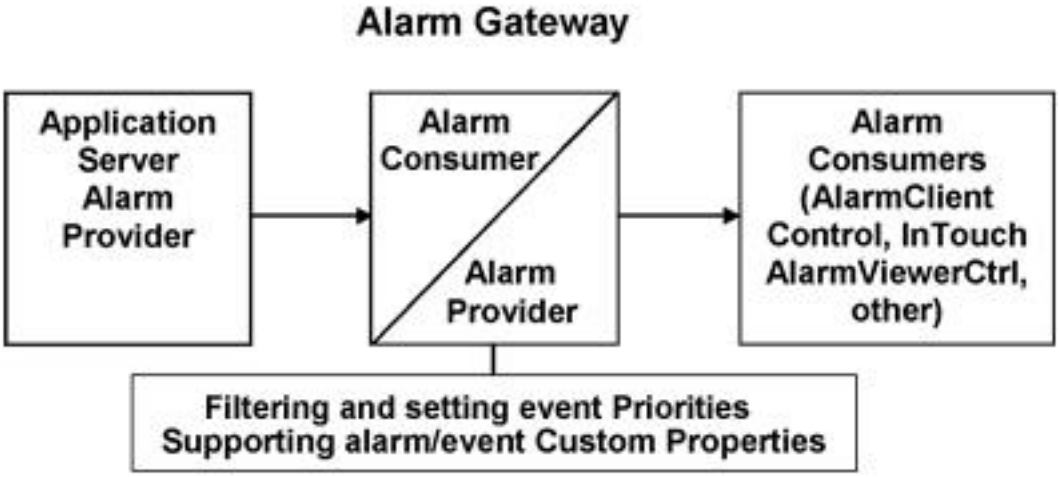
Table of Contents

Introduction 1
Installing the Alarm Gateway Object 3
Hardware requirements 3
Software requirements 3
Package content 4
Installing standalone object 5
Object import 5
Object configuration 6
Licensing requirements 13
Demo License installation 13
Software key installation 14
Configuration 15
General Configuration 15
Consumer: 15
Provider: 15
Run-Time Object Attributes 17
Custom Alarm/Event attributes 18
Wonderware alarm system custom attributes 19
Troubleshooting Wonderware alarm system custom attributes 26
Exposed Alarm Fields 33
Configuration attributes 37
AlarmIDList 37
Exposed attributes 39
Custom Acked alarm comment field 41
Overview 41
Configuration 43
UReason gateway 48
Alarm Gateway UReason Mimic functionality 49
Mimic functionally without UReason alarming system 51
Multiple distributed Alarm Gateway configuration setup sample 53
Object configuration 53
Alarm client configuration 59
Redundant Alarm Gateway configuration setup sample 61
Advanced Troubleshooting 67
Object upgrade procedure 69
Object clean uninstall procedure 70

Alarm Gateway Object for Wonderware Application Server

Introduction

The **Alarm Gateway Object** (Alarm Gateway) is a basic component of Wonderware Finland **Alarm Extension Pack Standard Edition** and provides functionality to create separate configurable Alarm Provider for alarms coming from Wonderware Application Server (WAS) and/or other Alarm Providers compatible with Wonderware Alarm System:



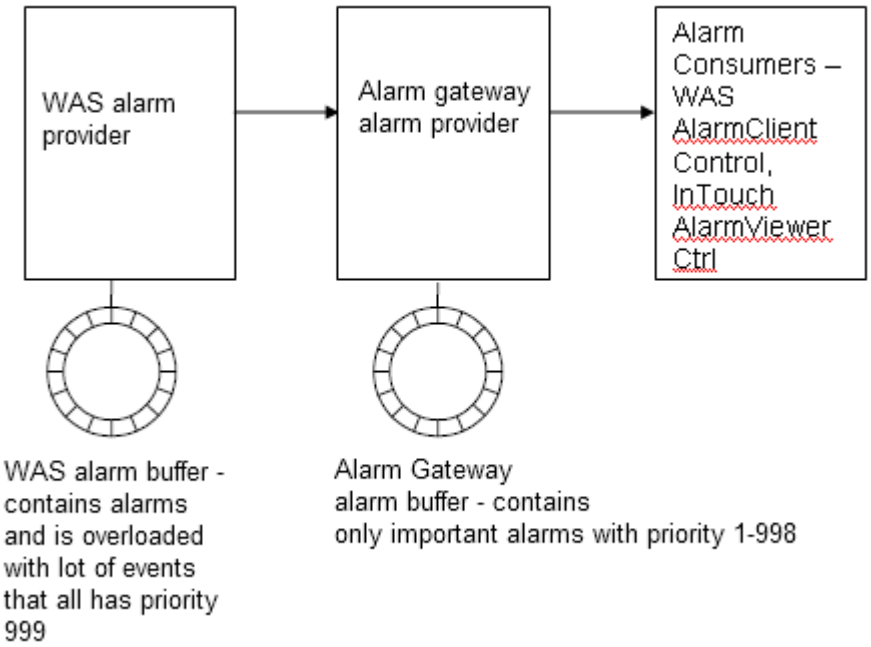
In case included in Wonderware Finland **Alarm Extension Pack UReason Edition**, the Alarm Gateway Object provides also the functionality to send/receive alarms to/from Wonderware alarming system from/to **UReason alarming system**.

The Alarm Gateway can be used to solve the following tasks:

<u>Avoid alarm loss in high loaded systems:</u>

WAS Historical alarms and events are stored in a circular buffer, where the oldest entries are discarded to make room for new ones, so in case there generated a lot of events then important alarms can be lost.

By using the Alarm Gateway, it is possible to store all important alarms in separate Alarm Gateway buffer - that can be done by querying alarms/events only with priorities from 1 to 998:



Note: Alarm Gateway alarm buffer can contain about 6000-7000 alarms. The total number of stored alarms depends on size of alarms.

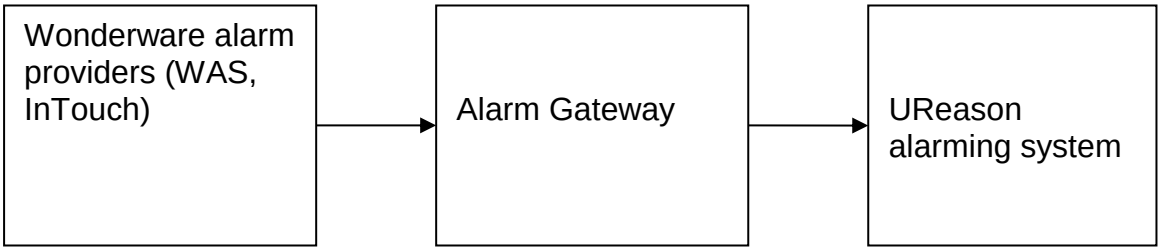
Change the event priority:

WAS alarming system does not provide possibility to configure event priority - all events have built-in priority 999.

By using Alarm Gateway, it is possible to change the event priority by using the setPriority custom attribute .For more information see the “Custom attributes” section “setPriority” later in this User Guide.

Connect to UReason alarming system:

Alarm Gateway can send alarms/events from Wonderware alarming system to UReason alarming system. For more information see “UReason gateway” section later in this User Guide.



Installing the Alarm Gateway Object

Hardware requirements

The Alarm Gateway object has the same hardware requirements as Wonderware Application Server. It is strongly recommended to have computer with 2 gigahertz (GHz) or faster processor, 32/64-bit. A multi-core processor is also strongly recommended. The Intel Itanium 2 processor is not supported.

Software requirements

The Wonderware **Application Server 3.1** version or later are supported.

Package content

- **AlarmGateway_001** folder **- contains** standalone **Alarm Gateway** object installation.

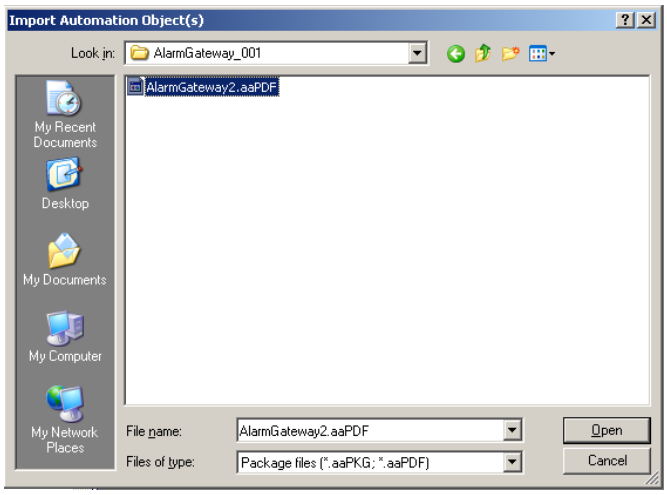
- **P185m22.pdf** – User Manual (this document)

Installing standalone object

Object import

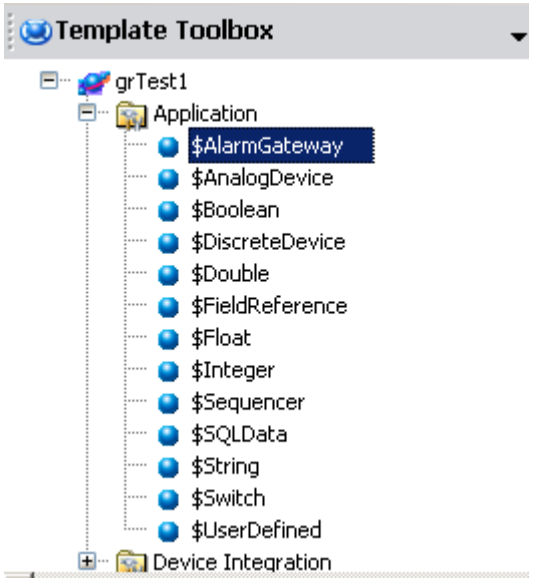
1) Copy **Alarm Gateway** object to some folder, e.g. to C:/Install.

2. Start the **ArchestrA IDE** and import the **AlarmGateway2.aaPDF** file from **AlarmGateway_001** folder to a new/existing galaxy (in the further explanation we will assume that a new galaxy **grTest1** is used).



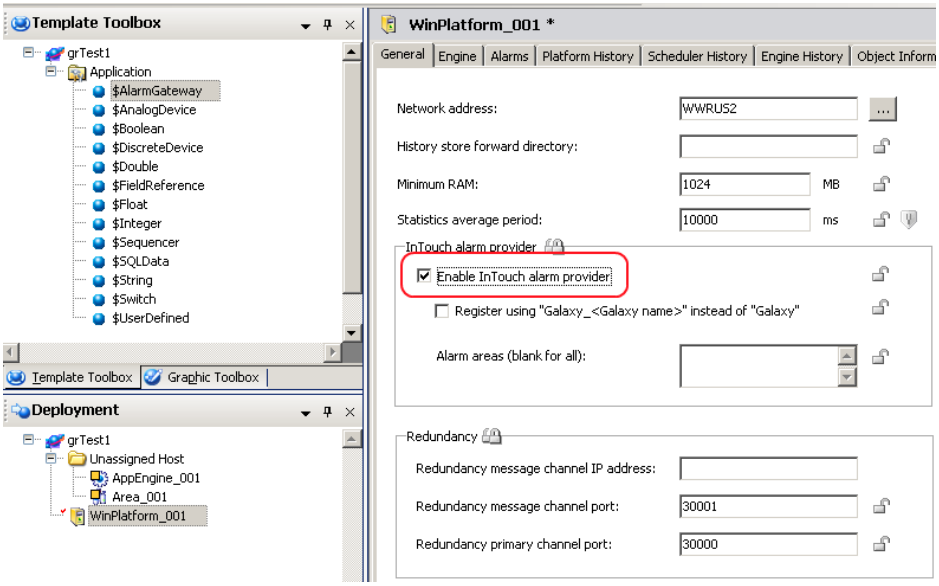
Note: If you are using existing galaxy and there are already deployed an older Alarm Gateway object version please, following upgrade instruction from section **Object upgrade procedure.**

3) After importing **$AlarmGateway** template is added to Template Toolbox:



Object configuration

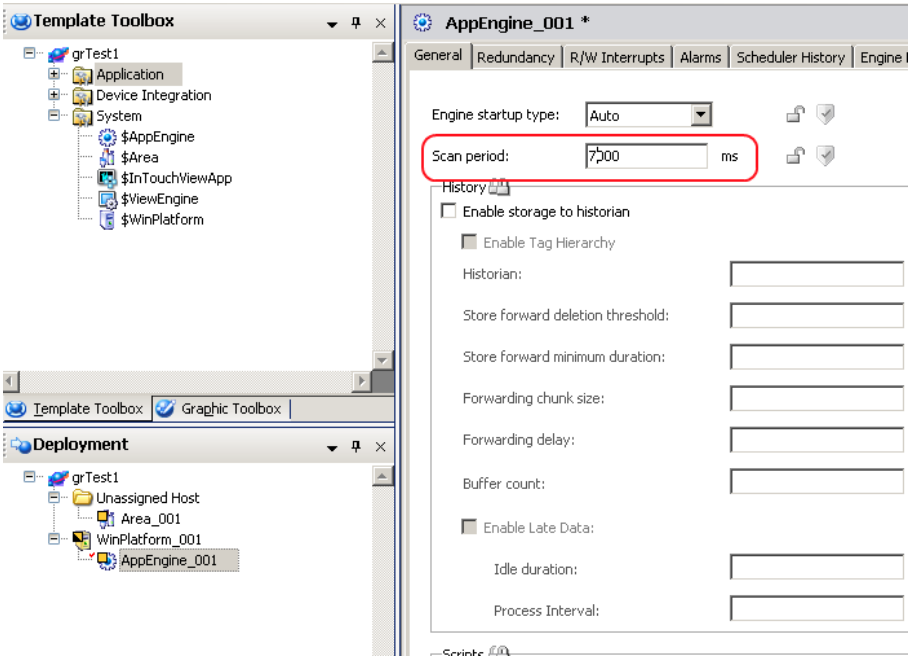
1) Create WinPlatform object (if is not existing already) with alarm provider feature enabled.

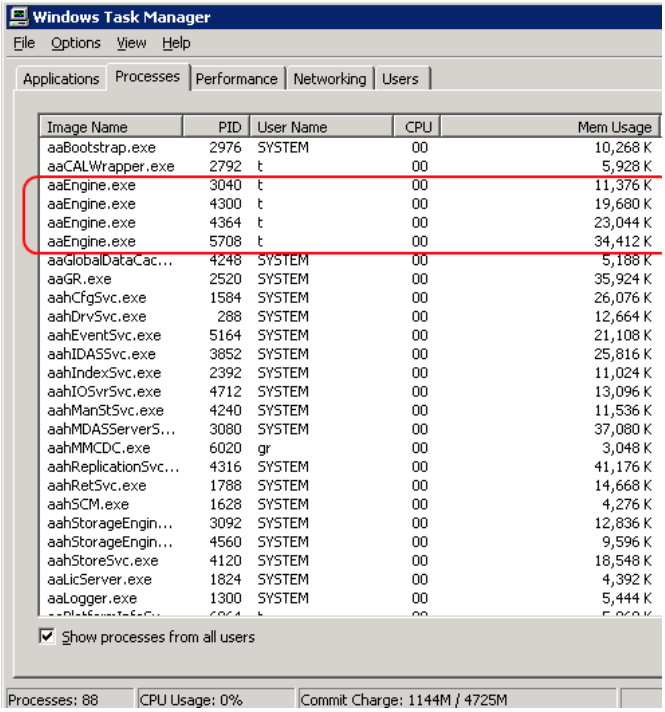


2) Create AppEngine and assign it to Platform object (Deployment View)

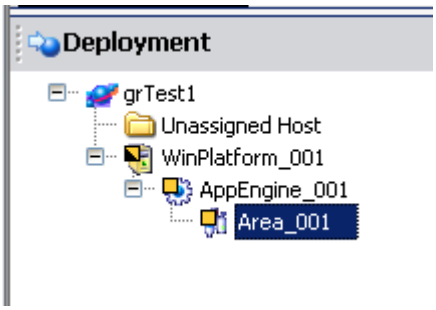
Note: Highly recommended is to deploy object Alarm Gateway to separate engine where are no production objects deployed, to distribute possible CPU load among multiple CPU cores.

Engine scan period is recommended to set at least 1 second and more (2-3 seconds) if object is planned to be used in high load alarm systems (more than 50 active alarms) and with enabled additional features like **Custom Alarm/Event attributes UReason alarm system**.





3) Create Area object and assign in to AppEngine object

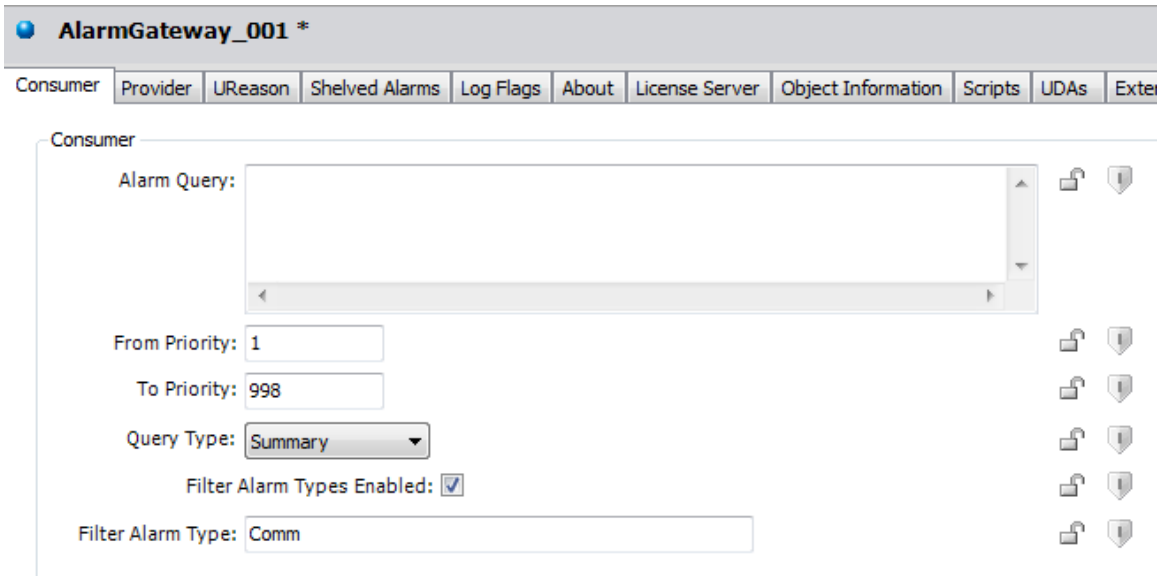


4) Create **Alarm Gateway** instance and assign it to Area object.
5) Open **Alarm Gateway** object editor and configure following parameters in:

a) Tab **Consumer**:

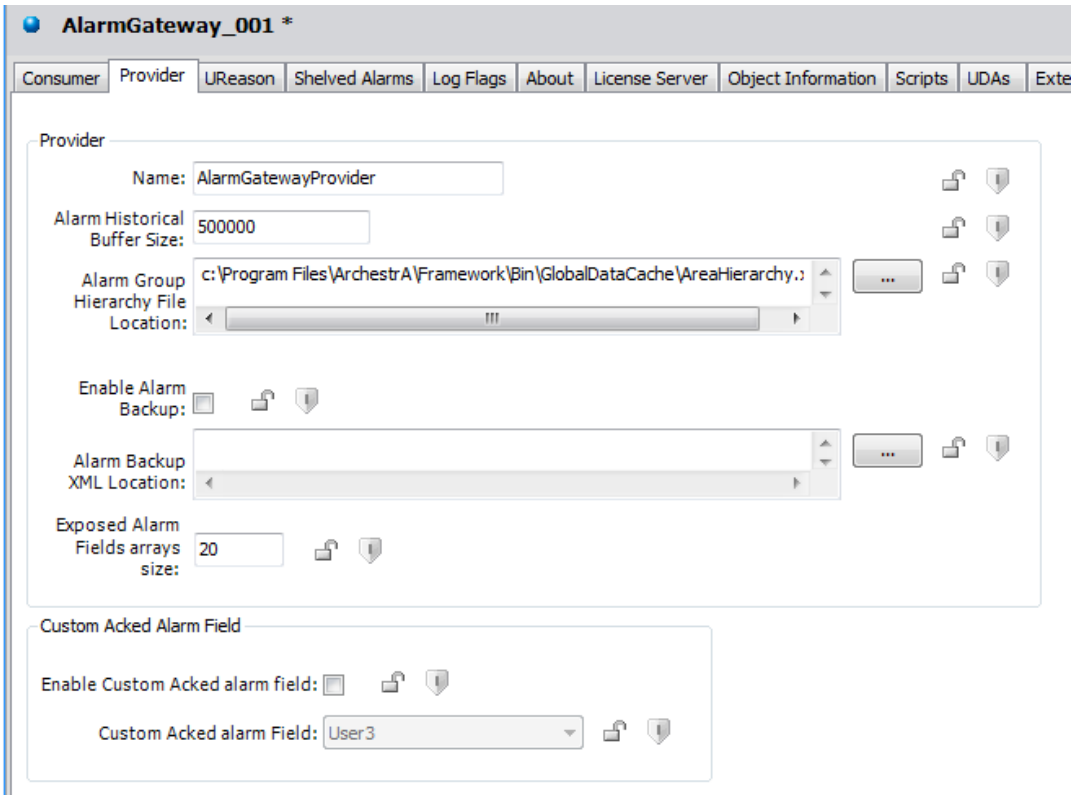
- Set **Alarm Query**: \Galaxy!Area_001
- Set **To Priority**: 1
- Set **From Priority**: 998
- Enable **Filter Alarm Types:** set filter Comm

Where **Area_001** is area name what is host of Alarm Gateway object or other area can be specified that host objects with alarming enabled.



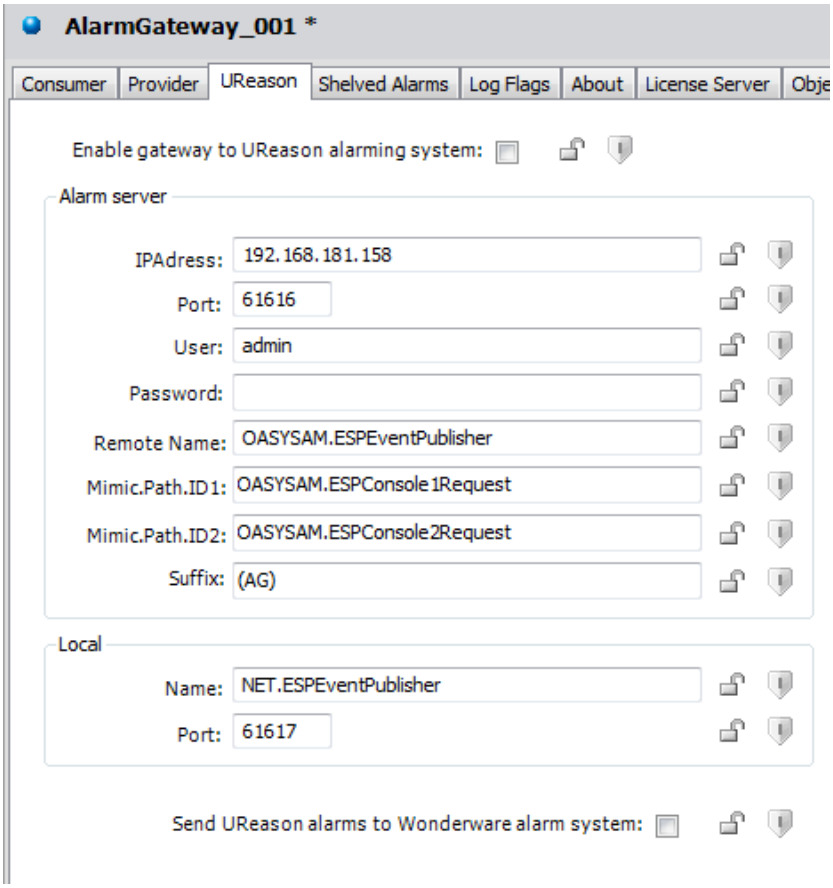
b) Tab **Provider**:

Change alarm provider name if needed of alarm clients:



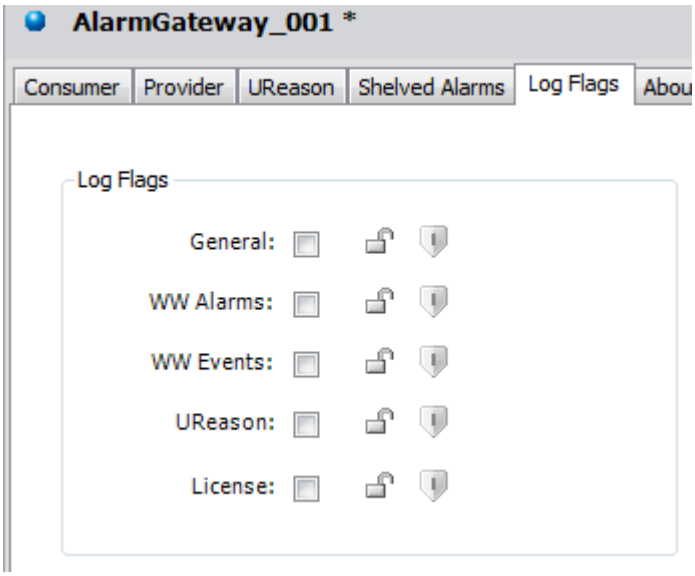
c) Tab **UReason**

Enable connection to **UReason** alarming system if used:



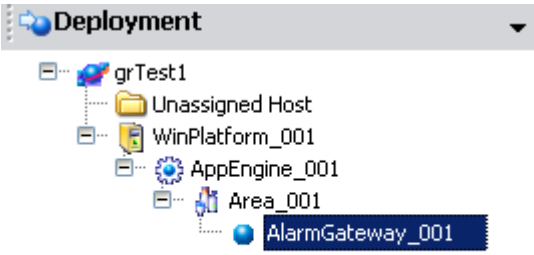
d) Tab **Log Flags**

If needed enable diagnostic logging, for more details see section **Advanced Troubleshooting**



6) Create or import some objects with alarming enabled under **Area_001**.

7) Deploy created all objects



8) License Alarm Gateway object - see Section **Licensing requirements** for details.

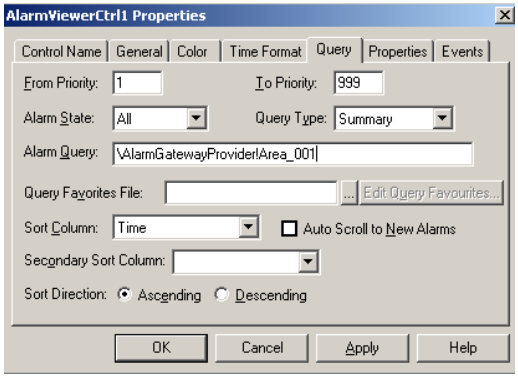
9) Create or import InTouch application to test AlarmGateway alarms – configure alarm client

Alarm Query:

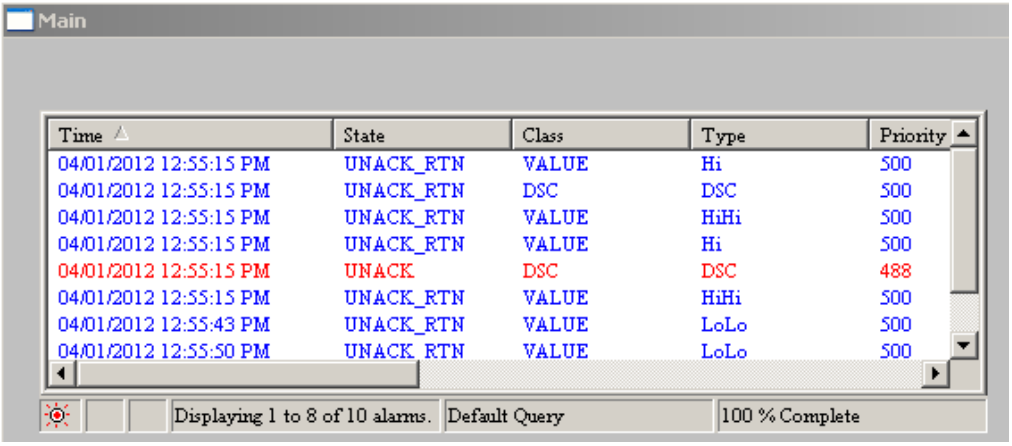
\\WWNode\AlarmGatewayProvider!Area_001 or

\\192.168.75.209\AlarmGatewayProvider!Area_001

Note: Query without Nodename like **AlarmGatewayProvider!Area_001** or **/AlarmGatewayProvider!Area_001** will not work on Windows 2008



10) Run InTouch application and check alarms



Time	State	Class	Type	Priority
04/01/2012 12:55:15 PM	UNACK_RTN	VALUE	Hi	500
04/01/2012 12:55:15 PM	UNACK_RTN	DSC	DSC	500
04/01/2012 12:55:15 PM	UNACK_RTN	VALUE	HiHi	500
04/01/2012 12:55:15 PM	UNACK_RTN	VALUE	Hi	500
04/01/2012 12:55:15 PM	UNACK	DSC	DSC	488
04/01/2012 12:55:15 PM	UNACK_RTN	VALUE	HiHi	500
04/01/2012 12:55:43 PM	UNACK_RTN	VALUE	LoLo	500
04/01/2012 12:55:50 PM	UNACK_RTN	VALUE	LoLo	500

Licensing requirements

Alarm Gateway object support two types of licenses:

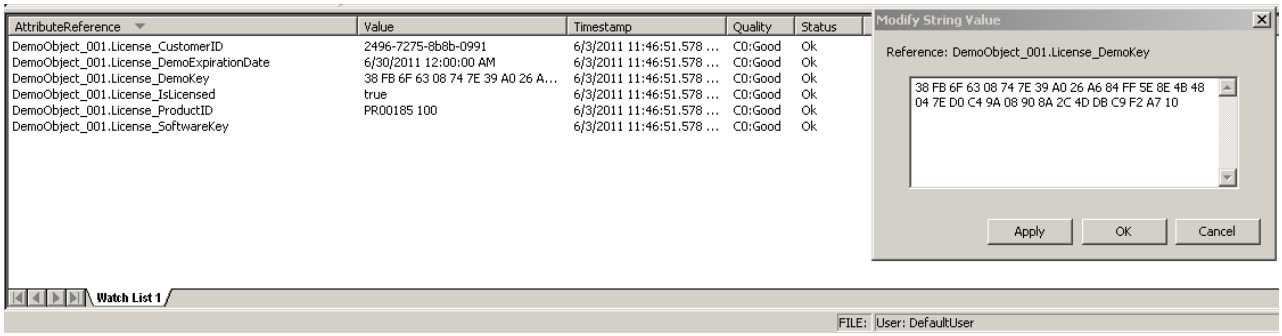
- The **demo license** is for free and provides an unlimited functionality, but it is valid only for a limited time period.
- The **software key** enables the Alarm Gateway Object unlimited full time running without any restrictions.

Demo License installation

The **demo license** is for free and provides an unlimited functionality, but it is valid only for a limited time period. After demo license expiration, the Alarm Gateway will stop to provide the alarms. The demo license can be obtained by sending inquiry to info@wonderware.fi.

To activate the received demo License key, you need to set it to object **License_DemoKey** attribute:

If demo license is valid (correct key is installed) **License_IsLicensed** attribute is true and in **License_DemoExpirationDate** attribute is displayed expiration date after that product stops working.



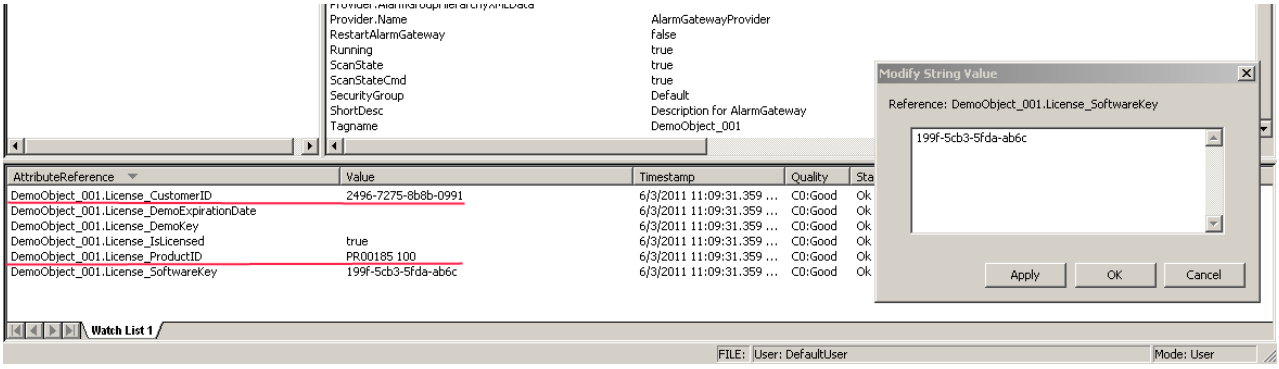
Software key installation

The **software key** enables the Alarm Gateway Object unlimited full time running without any restrictions.

To get and enable the **software key**:

- get “Product ID” from object **License_ProductID** attribute (e.g. PR00185 100);
- get “Customer ID” from object **License_CustomerID** attribute;
- copy/paste it to e-mail (or text file or similar) and provide this “Customer ID” string when ordering the Alarm Gateway Object;
- when product is purchased, copy the received “Software Key” to object **License_SoftwareKey** attribute:

If license key is valid (correct key is installed) **License_IsLicensed** attribute is set to true and product is ready for use.



Licensing run-time attributes:

Attribute	Description	Run-Time Access
License_CustomerID	Unique generated customer ID	Read-Only
License_DemoExpirationDate	Demo license expiration date	Read-Only
License_DemoKey	Demo license key	User
License_IsLicensed	If True then product is licensed	Read-Only
License_ProductID	Product ID	Read-Only
License_SoftwareKey	Product software key	User

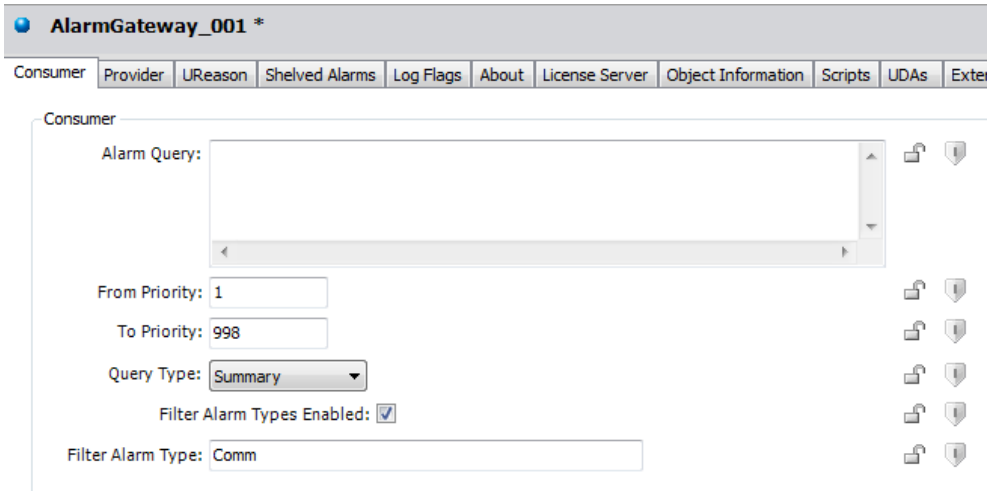
Configuration

For general information about objects (including relationships, deployment and alarm distribution) - see the Wonderware Integrated Development Environment (IDE) documentation.

For information on configuration options for object information, scripts, user-defined attributes (UDAs), or attribute extensions, click **Extensions Help** in the Help file header.

General Configuration

Use the **Consumer** tab to configure and tune the behavior of the Alarm Gateway Object consumer:

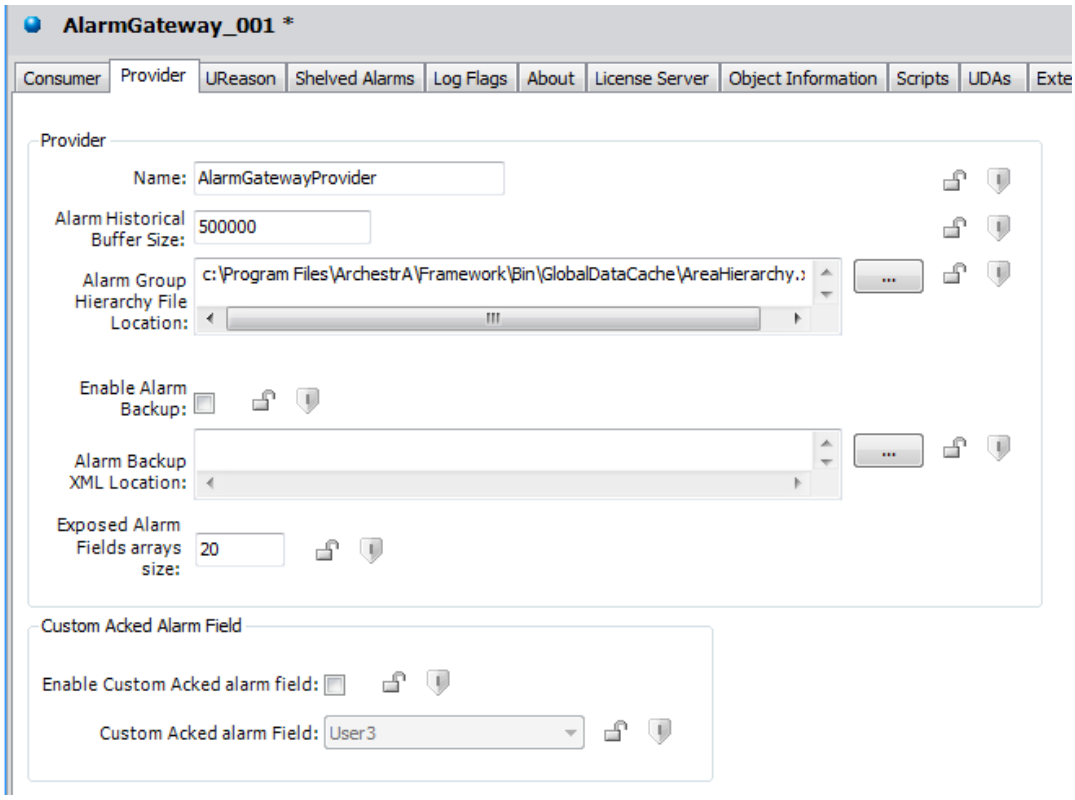


Consumer:

Editor Option	Associated Attribute (s)	Description
Alarm Query	Consumer.AlarmQuery	Consumer Alarm Query
From Priority	Consumer.FromPriority	Enter the starting value of the alarm priority range
To Priority	Consumer.ToPriority	Enter the ending value of the alarm priority range
Query Type	Consumer.QueryType	Alarm query type.
Alarm Type filter	Enable/disable: Consumer.FilterAlarmTypesEnabled Consumer.FilterAlarmTypes	Provider advanced functionality to filter alarms by type e.g. can filter out all Comm alarms.

Use the **Provider** tab to configure and tune the behavior of the Alarm Gateway Object **provider**:

Provider:



Editor Option	Associated Attribute	Description
Name	Provider.Name	Alarm provider name
Alarm Historical Buffer Size	Provider.AlarmBufSize	Alarm buffer size.
Alarm Group Hierarchy XML	Provider.AlarmHierarchyFile	Path to WAS generated Alarm (Area) hierarchy file Default value: c:\Program Files\ArchestrA\Framework\Bin\GlobalDataCache\AreaHierarchy.xml
Alarm Backup XML Location	Provider.AlarmBackupLocation	Alarm Backup XML files Location on disk
Exposed Alarm Fields arrays Size	Exposed_Alarm_Fields_Arrays_Size	Exposed Alarm Fields array size for more details see section **Exposed Alarm Fields**
Custom Acked Alarm field	Enable/disable: EnableCustomAckField CustomAckAlarmField	For more details see section **Custom Acked alarm comment field**

Run-Time Object Attributes

All object attributes are grouped into following groups by prefix:

AlarmGateway - defines attributes for Alarm Gateway general configuration and status**.**
Provider_ - defines attributes for Alarm Provider configuration
Consumer_ - defines attributes for Alarm Consumer configuration.
Licence_ - defines attributes for licensing
Set - defines custom attributes – see section “Custom attributes” for more information

The following table describes the run-time only attributes for the Alarm Gateway Object.

Note: Configurable run-time attributes are described in the configuration sections. For more information, see **Configuration** section above.

Attribute	Description	Run-Time Access
AlarmGateway_Started	If true Alarm Gateway is Started and running.	Read-Only
AlarmGateway_LastErrorMessage	Last Error Message	Read-Only
AlarmGateway_LastErrorCode	Last Error Code (No errors = 0)	Read-Only
AlarmGateway_Restart	Trigger – if set to True then restarts Alarm Gateway.	User
Consumer.Status	Current status of Alarm Gateway	Read-Only
AlarmGateway_AlarmGroups	Displays all created alarm groups	Read-Only

Note: It is highly recommended to run any Alarm Gateway Object in separate Engine since Alarm Gateway uses scan interval for reading the alarms. Recommended Engine scan interval for Alarm Gateway is at least 1000 ms.

Custom Alarm/Event attributes

By using custom attributes, it is possible to change following alarm data fields in Wonderware alarm system or in UReason alarm system:

For Wonderware alarm custom attributes prefix **setWW**_ is used for Ureason attribute prefix **setUR**_ is used.

Custom Attribute	Alarming system	Description
SetWW_User1	Wonderware	User-defined field number 1.
SetWW_User2	Wonderware	User-defined field number 2.
SetWW_User3	Wonderware	User-defined field, string.
SetWW_Priority	Wonderware	Alarm/Event Priority.
SetUR_Source	UReason	Alarm Source
SetUR_Class	UReason	Alarm Class

Custom attributes can be set from WAS scripts with following command:

Syntax: objectName.CustomAttribute = “Alarm/Event name = value”

Wonderware alarm system custom attributes

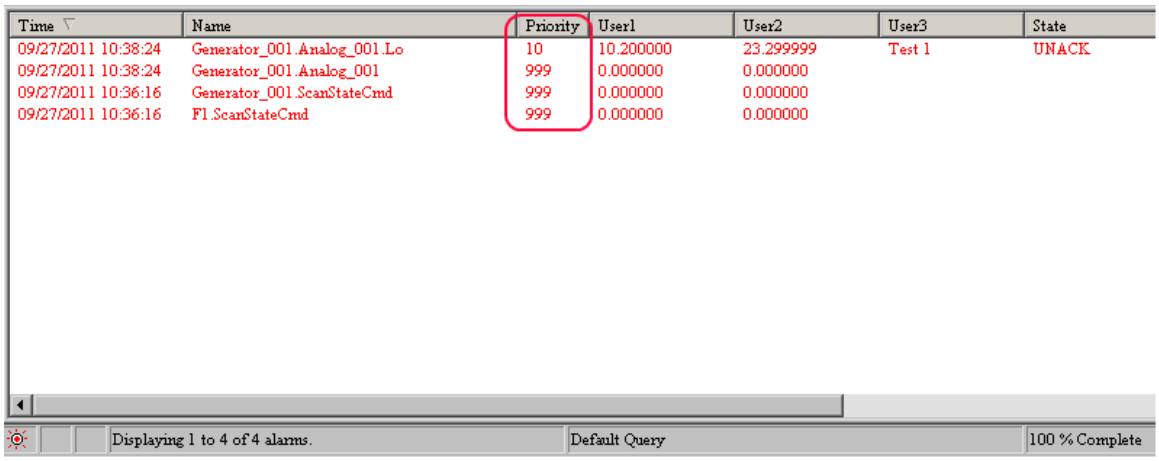
SetWW_Priority

Is used to set **Wonderware** alarm system alarm and event **Priority** (valid range from 1 to 999).

Sample:

Following command sets Wonderware alarming system alarm priority to 10 for alarm Generator_001.Analog_001.Lo:

```
AlarmGateway_001.setWWPriority =  Me.Tagname + ".Analog_001.Lo=10";
```



Time	Name	Priority	User1	User2	User3	State
09/27/2011 10:38:24	Generator_001.Analog_001.Lo	10	10.200000	23.299999	Test 1	UNACK
09/27/2011 10:38:24	Generator_001.Analog_001	999	0.000000	0.000000		
09/27/2011 10:36:16	Generator_001.ScanStateCmd	999	0.000000	0.000000		
09/27/2011 10:36:16	F1.ScanStateCmd	999	0.000000	0.000000		

Displaying 1 to 4 of 4 alarms. | Default Query | 100 % Complete

Note: value is needed to set before alarm is active.

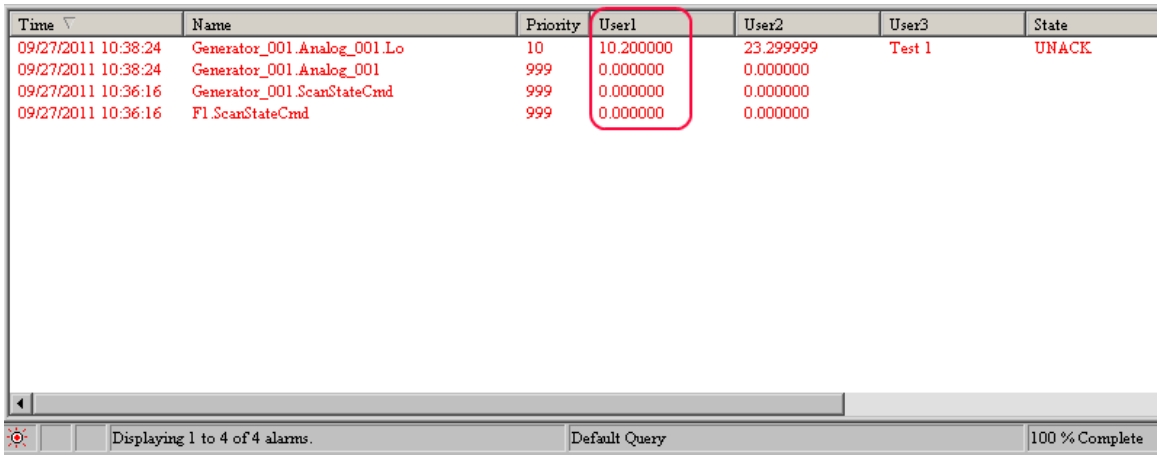
SetWW_User1

Is used to set **Wonderware** alarm system User-defined **(User1)** float field.

Sample:

Following command sets Wonderware alarming system alarm User 1 field to 10.2 for alarm Generator_001.Analog_001.Lo:

```
AlarmGateway_001.setWW_User1 =  Me.Tagname + ".Analog_001.Lo=10.2";
```



Time	Name	Priority	User1	User2	User3	State
09/27/2011 10:38:24	Generator_001.Analog_001.Lo	10	10.200000	23.299999	Test 1	UNACK
09/27/2011 10:38:24	Generator_001.Analog_001	999	0.000000	0.000000		
09/27/2011 10:36:16	Generator_001.ScanStateCmd	999	0.000000	0.000000		
09/27/2011 10:36:16	F1.ScanStateCmd	999	0.000000	0.000000		

Displaying 1 to 4 of 4 alarms. | Default Query | 100 % Complete

Note: value is needed to set before alarm is active.

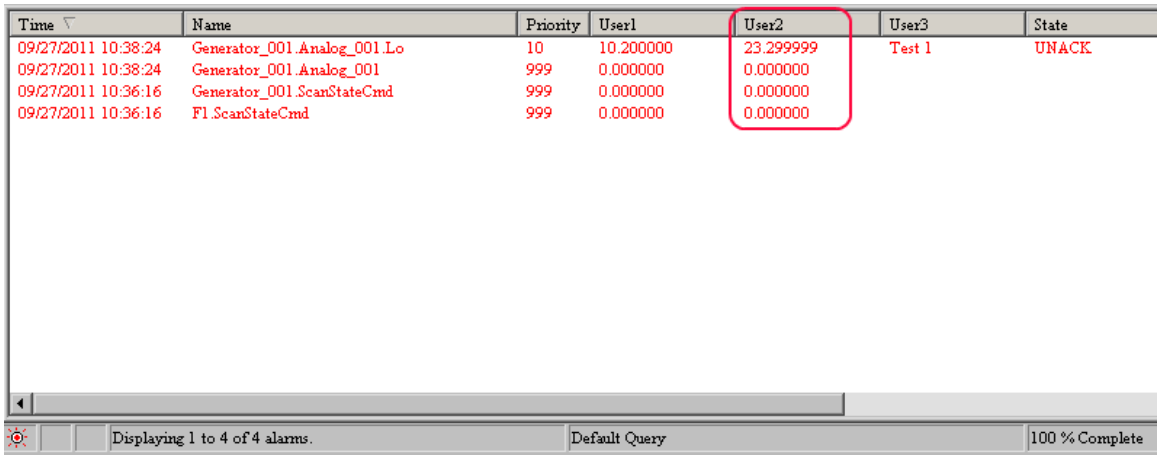
SetWW_User2

Is used to set **Wonderware** alarm system User-defined **(User2)** float field.

Sample:

Following command sets Wonderware alarming system alarm User 2 field to 23.3 for alarm Generator_001.Analog_001.Lo:

```
AlarmGateway_001.setWW_User2 =  Me.Tagname + ".Analog_001.Lo=23.3";
```



Time	Name	Priority	User1	User2	User3	State
09/27/2011 10:38:24	Generator_001.Analog_001.Lo	10	10.200000	23.299999	Test 1	UNACK
09/27/2011 10:38:24	Generator_001.Analog_001	999	0.000000	0.000000		
09/27/2011 10:36:16	Generator_001.ScanStateCmd	999	0.000000	0.000000		
09/27/2011 10:36:16	F1.ScanStateCmd	999	0.000000	0.000000		

Displaying 1 to 4 of 4 alarms. | Default Query | 100 % Complete

Note: value is needed to set before alarm is active.

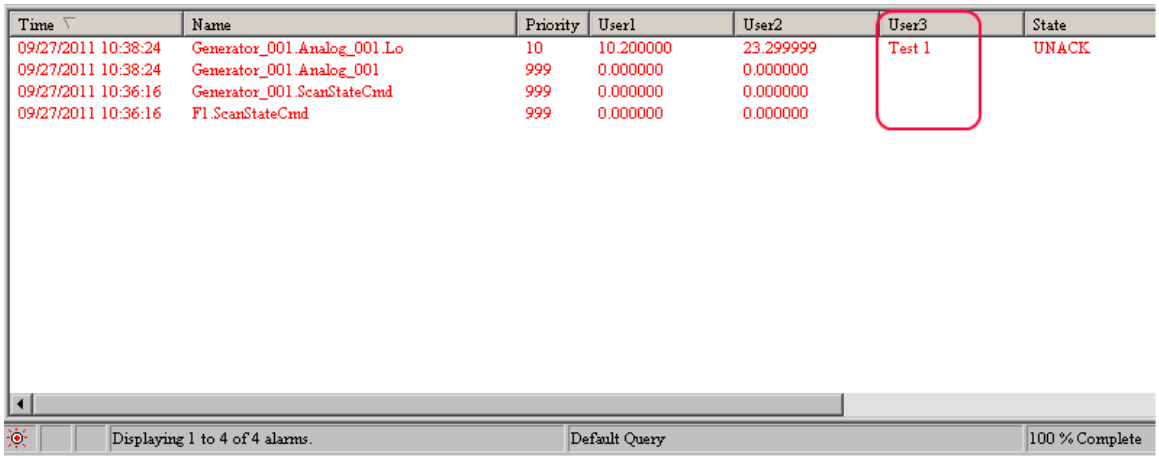
SetWW_User3

Is used to set **Wonderware** alarm system User-defined **(User3)** string field.

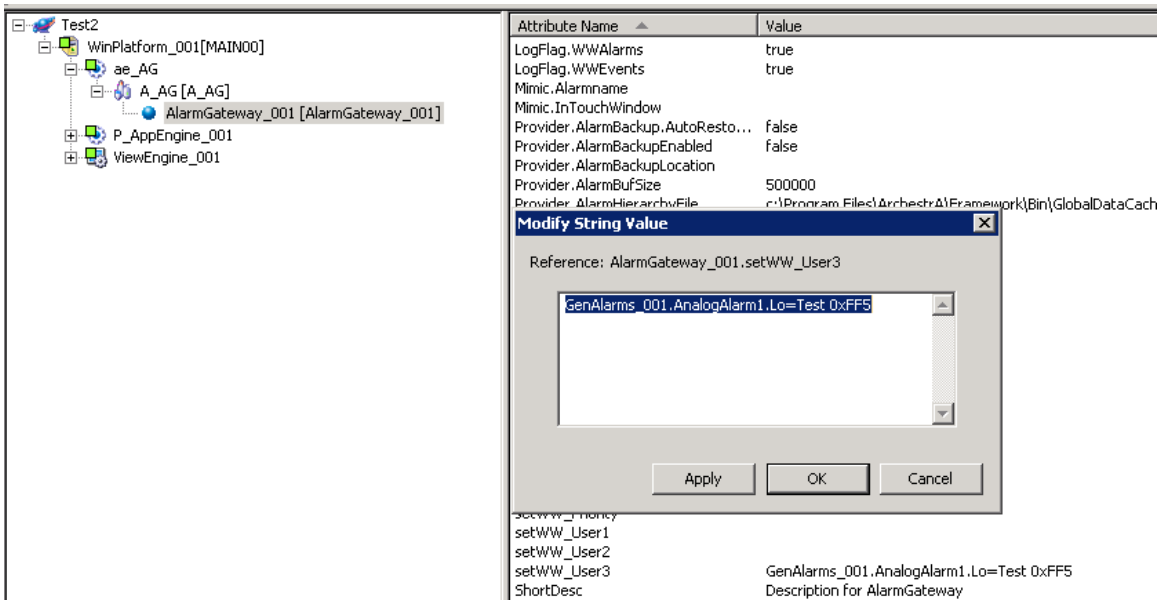
Sample:

Following command sets Wonderware alarming system alarm User 3 field to 'Test 1' for alarm Generator_001.Analog_001.Lo:

```
AlarmGateway_001.setWW_User3 =  Me.Tagname + ".Analog_001.Lo=Test 1";
```



Setting value from Object Viewer:



Note: value is needed to set before alarm is active.

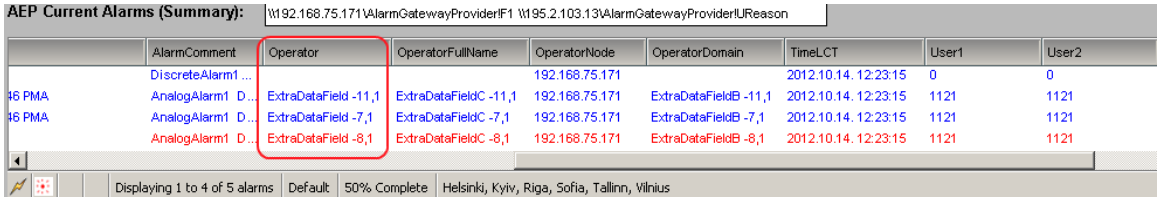
SetWW_Operator

Is used to set **Wonderware** alarm system User-defined **(**Operator**)** string field.

Sample:

Following command sets Wonderware alarming system alarm Operator field to ‘Test 1’ for alarm Generator_001.Analog_001.Lo:

```
AlarmGateway_001.setWW_Operator =  Me.Tagname + ".Analog_001.Lo=Test 1";
```



Note: value is needed to set before alarm is active.

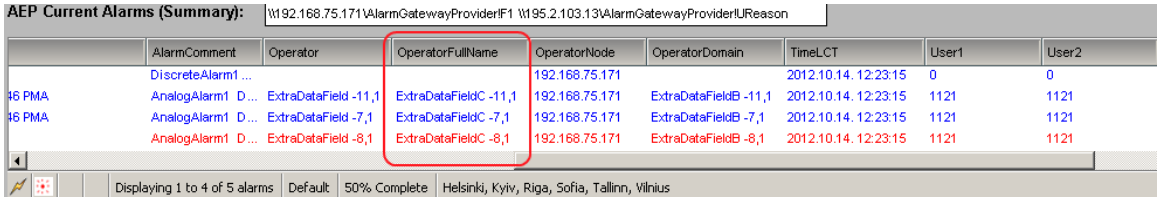
SetWW_OperatorFullName

Is used to set **Wonderware** alarm system User-defined **(**OperatorFullName**)** string field.

Sample:

Following command sets Wonderware alarming system alarm OperatorFullName field to ‘Test 1’ for alarm Generator_001.Analog_001.Lo:

```
AlarmGateway_001.setWW_OperatorFullName =  Me.Tagname +
".Analog_001.Lo=Test 1";
```



Note: value is needed to set before alarm is active.

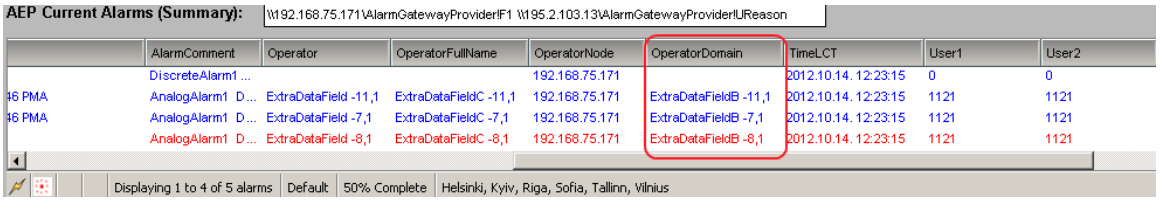
SetWW_OperatorDomain

Is used to set Wonderware alarm system User-defined (OperatorDomain) string field.

Sample:

Following command sets Wonderware alarming system alarm Operator Domain field to 'Test 1' for alarm Generator_001.Analog_001.Lo:

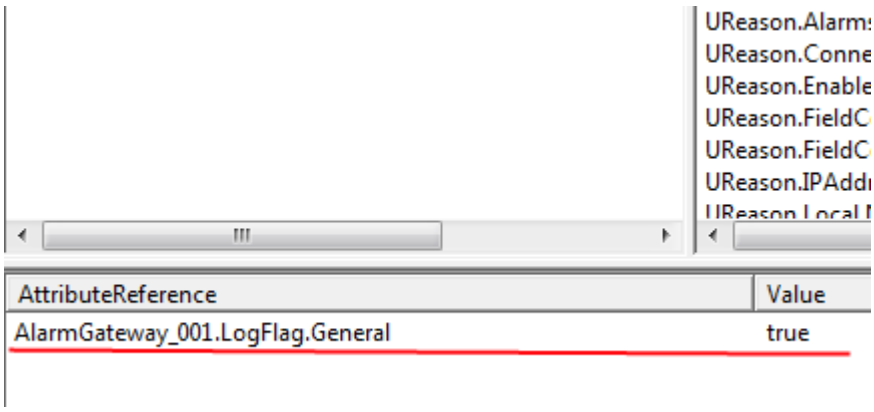
```
AlarmGateway_001.setWW_ OperatorDomain = Me.Tagname +
".Analog_001.Lo=Test 1";
```



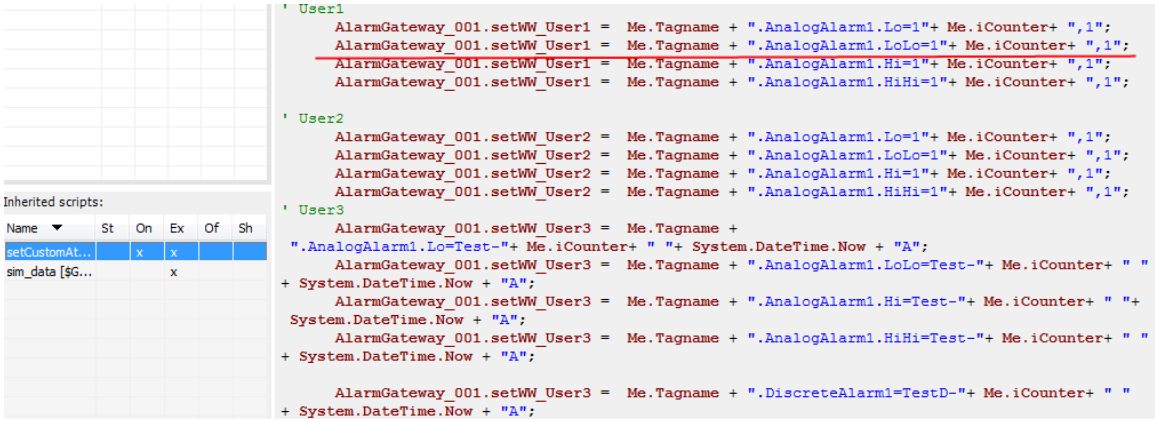
Note: value is needed to set before alarm is active.

Troubleshooting Wonderware alarm system custom attributes

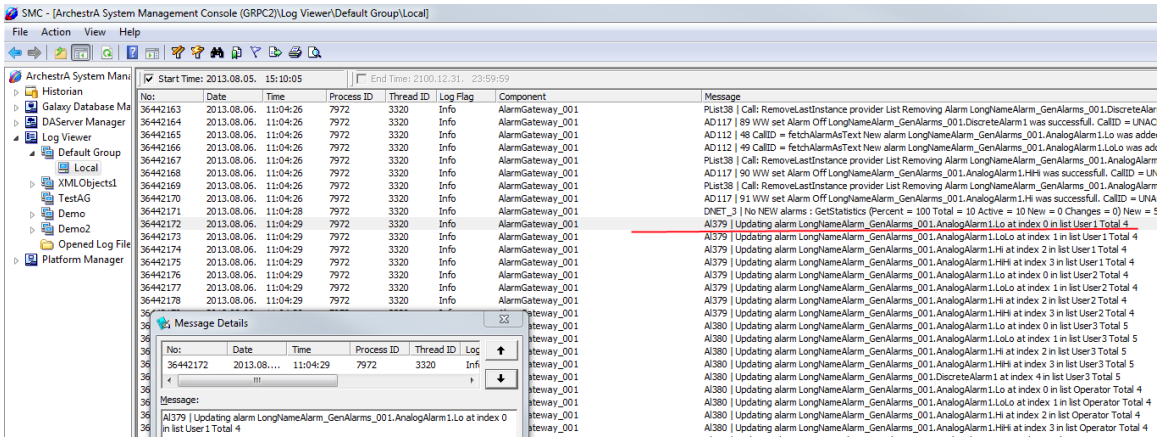
In case custom attribute is not set for alarm following steps can be performed to troubleshoot the issue.
Enable general alarm flag:



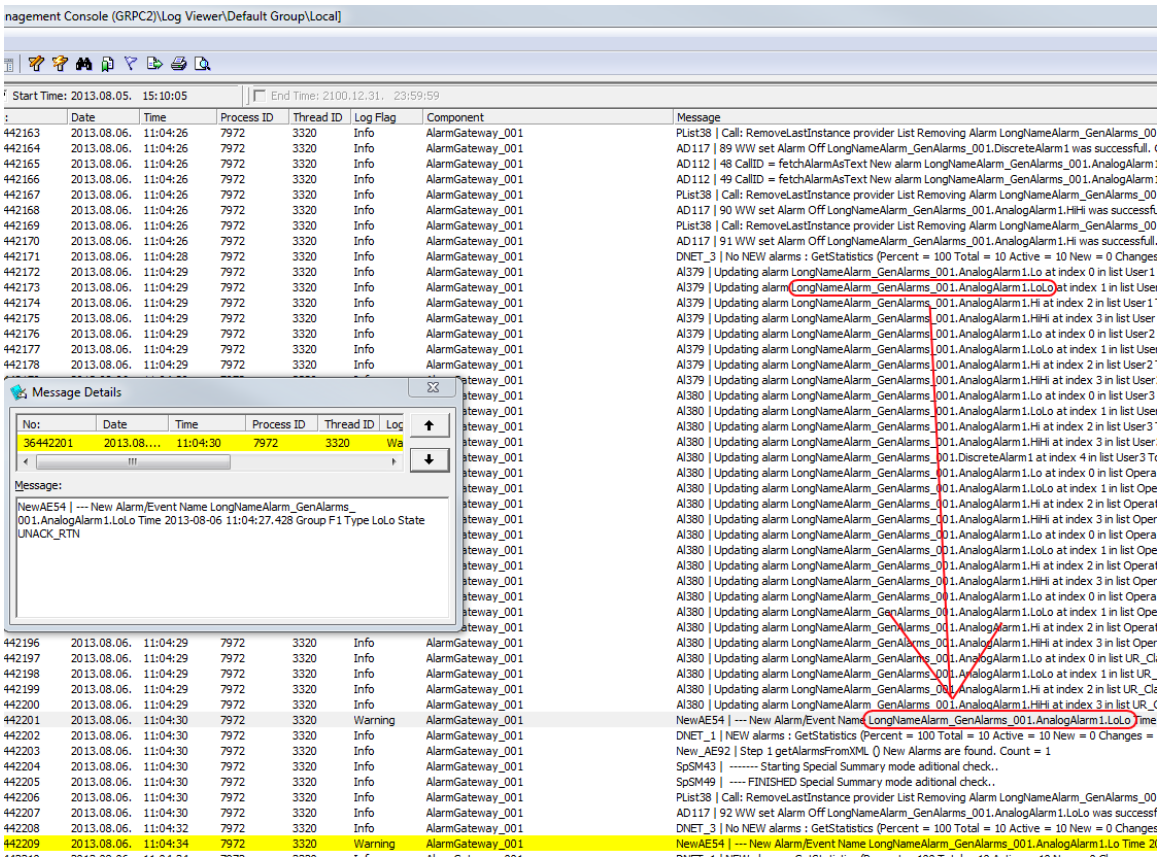
Set custom attribute e.g. from WAS script:



After script is executed check is appropriate message (**AI379**) logged in logger that indicates that custom attribute is set:



Rise alarm and check is Alarm name is the same as alarm name defined in set command:



Alarm Name mentioned in messages Al379 and NewAE54 must match to function work properly.

UReason alarm system custom attributes

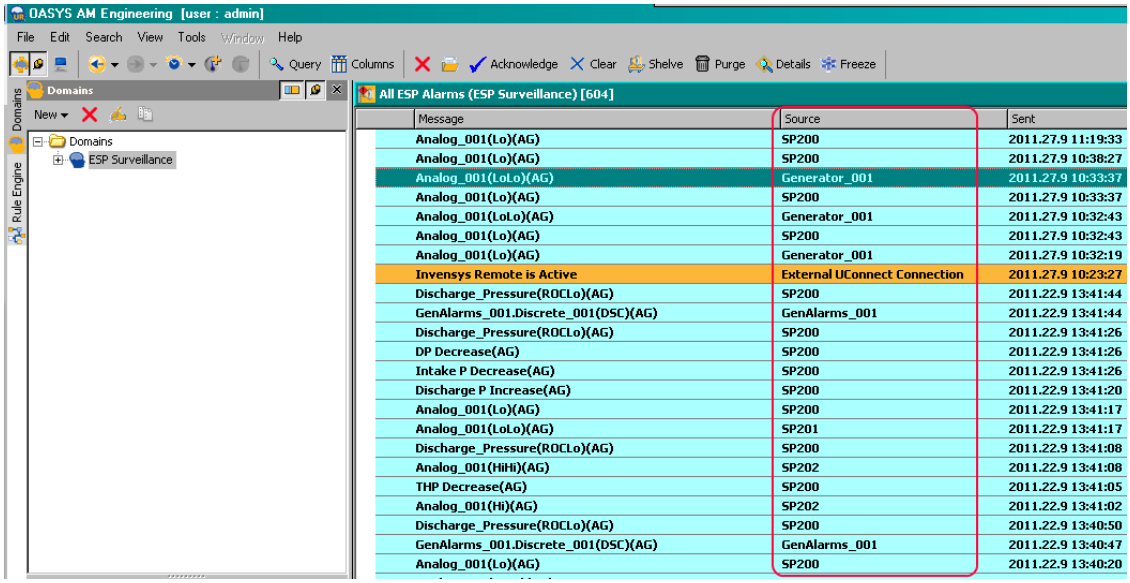
setUR_Source

Is used to set **UReason** alarm system alarm **Source** property.

Sample:

Following command sets UReason alarm parameter Source to ‘SP200’ for alarm Generator_001.Analog_001.Lo:

```
AlarmGateway_001.setURSource = Me.Tagname + ".Analog_001.Lo=SP200";
```



setUR_Class

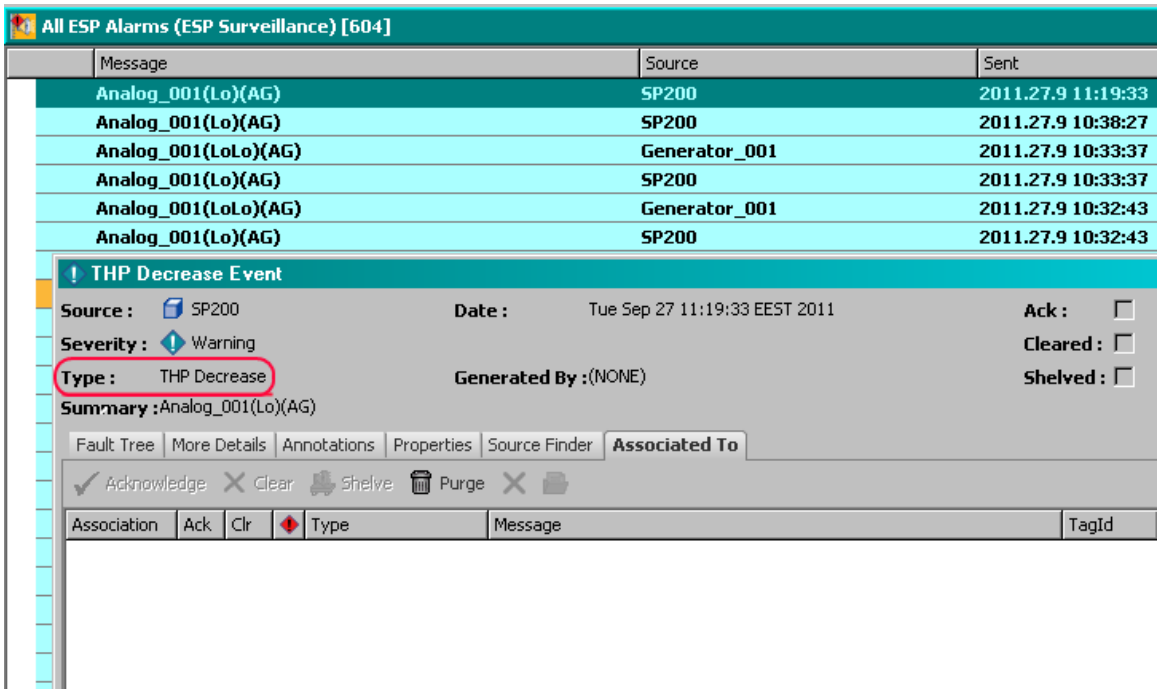
Is used to set **UReason** alarm system alarm **Type** property.

Note: Class is a critical parameter for UReason alarm system, all alarms/events that are intended for use in UReason alarm system must have defined valid class.

Sample:

Following command sets UReason alarm Class (Type) property to 'THP Decrease' for alarm Generator_001.Analog_001.Lo:

```
AlarmGateway_001.setURClass = Me.Tagname + ".Analog_001.Lo=THP Decrease";
```



setUR_Priority

Is used to set **UReason** alarm system alarm **Severity** property.

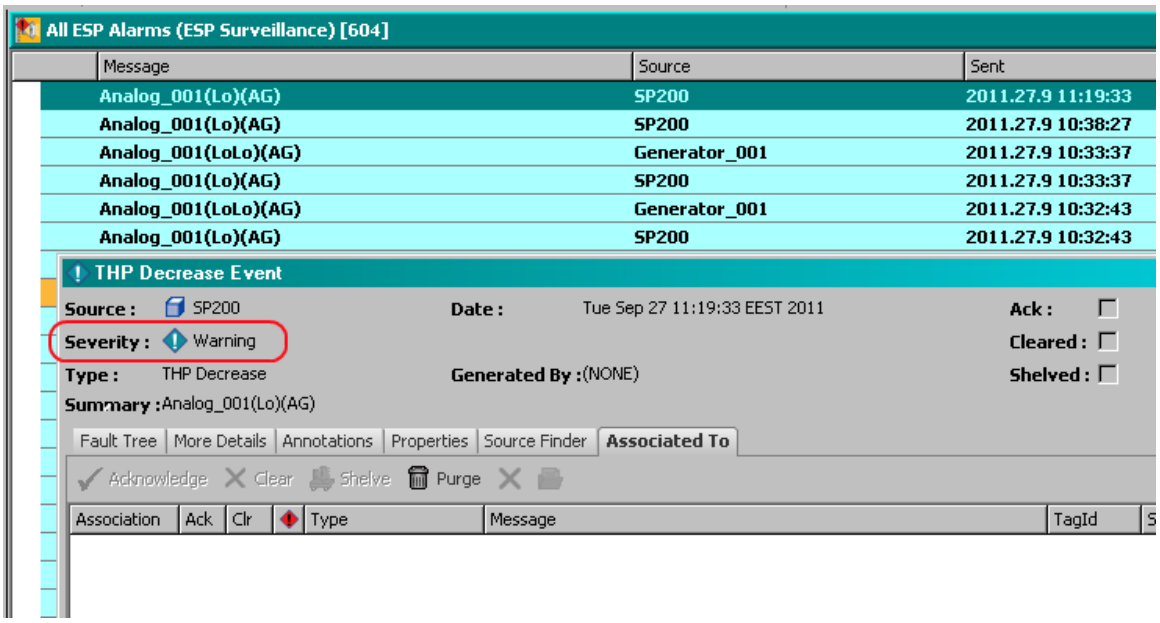
Sample:

Following command sets UReason alarm **Severity** property to 5 (Warning) for alarm Generator_001.Discrete_001:

```
AlarmGateway_001.setUR_Priority =  Me.Tagname + ".Discrete_001=5";
```

Note: UReason alarm system has following alarm/event priorities:

1 - Critical
2 – High Severity
3 – Medium Severity
4 - Low Severity
5 - Warning
6 – Information



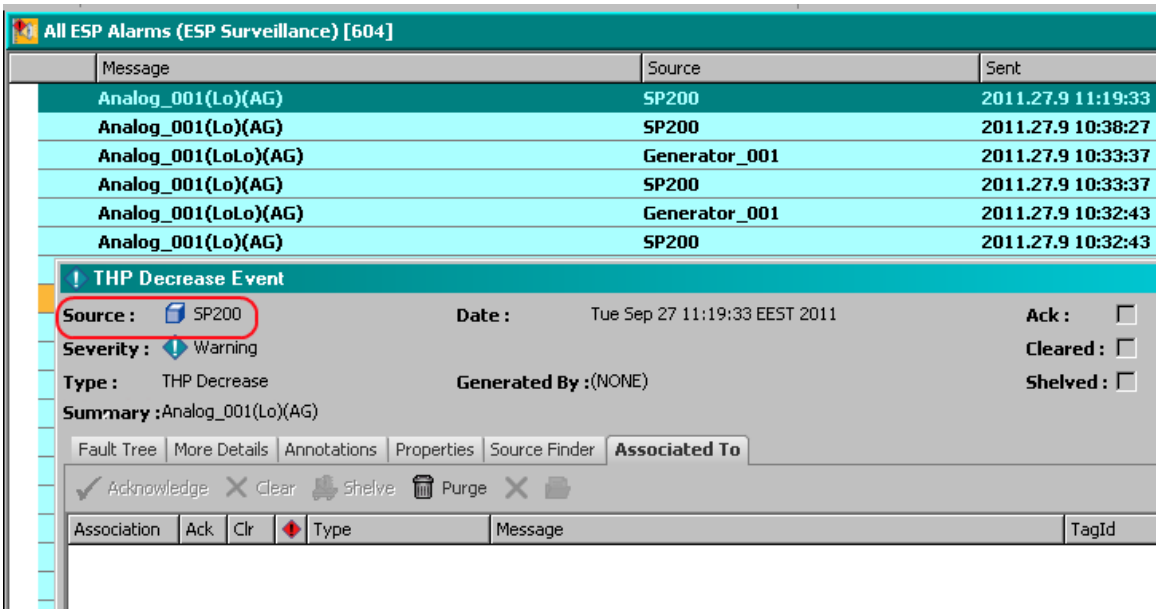
setUR_Source

Is used to set **UReason** alarm system alarm **Source** property.

Sample:

Following command sets UReason alarm **Source** property to Me.Tagname (Generator_001) value for alarm Generator_001.Analog_001.Lo:

```
AlarmGateway_001.setUR_Source = Me.Tagname + ".Analog_001.Lo=" + Me.Tagname;
```



setUR_MimicWindow

Is used to set **UReason** alarm system alarm **MimicWindow** property.

For details see **Alarm Gateway UReason Mimic functionality** section.

Sample:

Following command sets **UReason** alarm MimicWindow property:

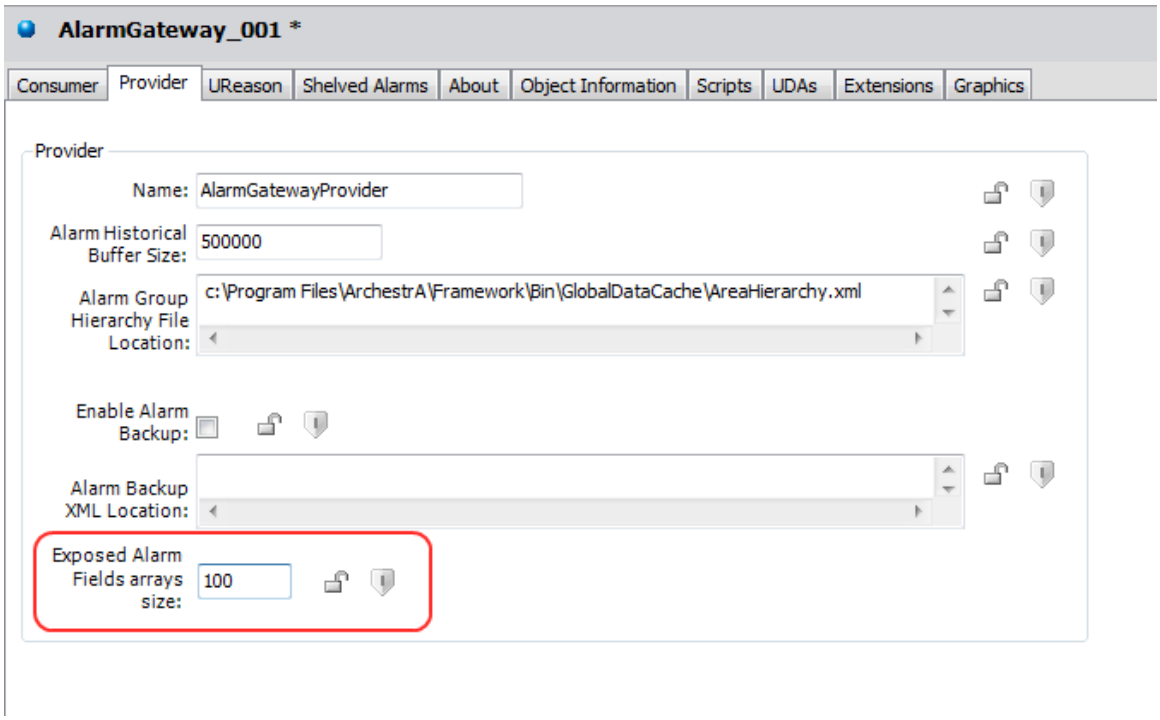
```
AlarmGateway_001.setUR_MimicWindow =  Me.Tagname + ".Analog_001.Lo=SP200";
```

Exposed Alarm Fields

Exposed Alarm Field arrays are used to expose specific Alarm information (fields) as object attributes that can be used in WAS scripting.

Following configuration is needed to setup Exposed Alarm Fields:

1) In object editor open tab Provider
   a) Enable Exposed Alarm Fields feature.
   b) and set Exposed Alarm Fields arrays length:



   c) Deloy **Alarm Gateway** object.

   d) Define exposed alarm configuration XML:

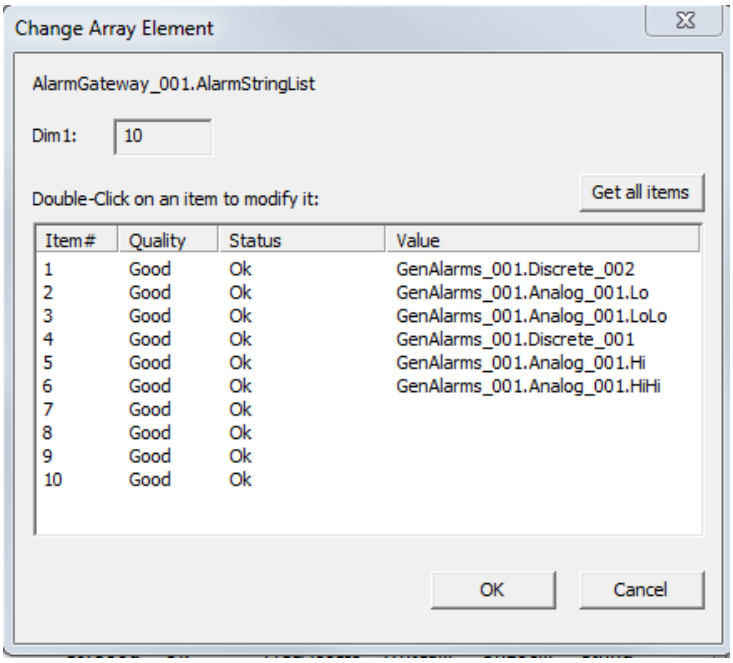
XML structure:
Root: <AlarmConfiguration>
Element:
Name: Alarm

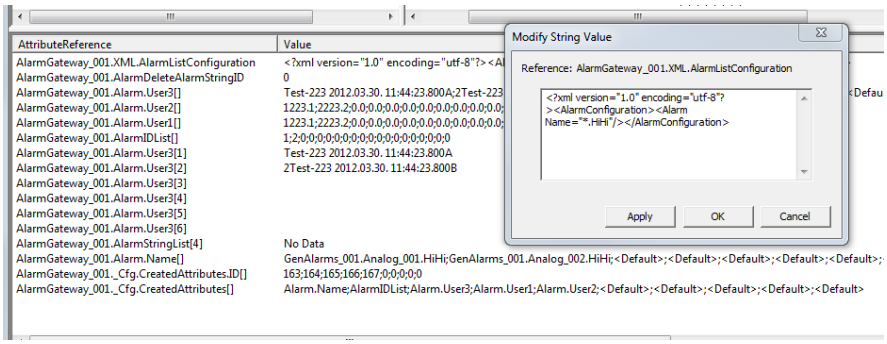
Attributes:

- Name – Alarm name – **alarm names are case sensitive.**
  Note: Also wildcards are supported following samples are correct:

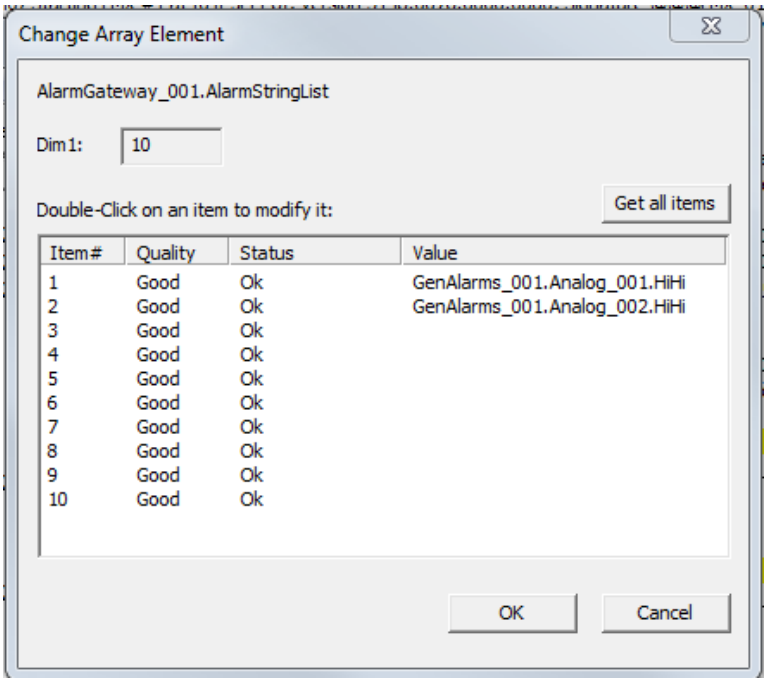
  1) GenAlarm*

  Adds to exposed list all Alarms that name starts with GenAlarm:

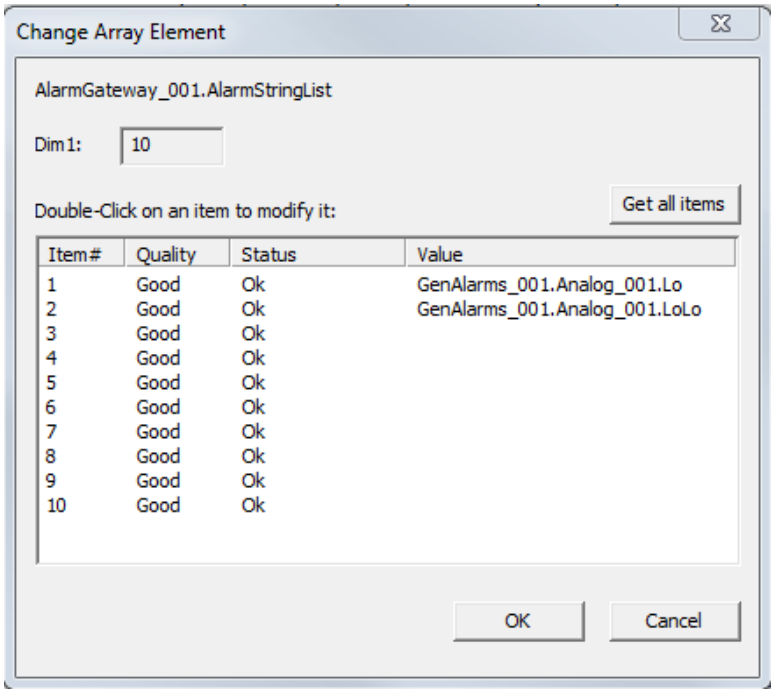




2) *HiHi - Adds to exposed list all Alarms that name ends with HiHi – all HiHi priority alarms.



3) *Value1.Lo*



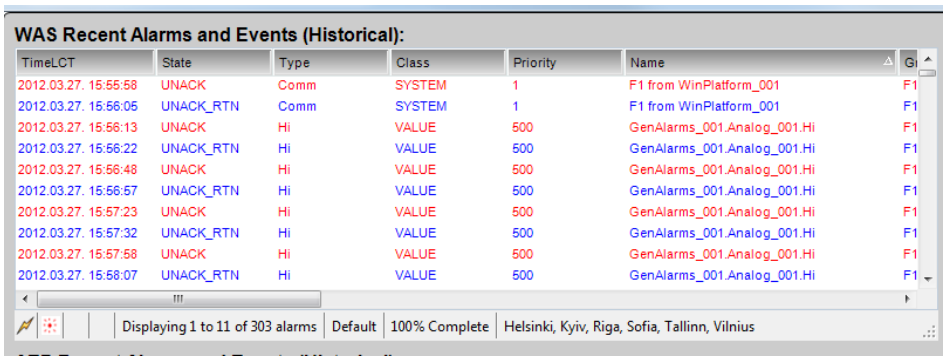
- ID – used defined ID -is used to delete entries from arrays by set value to **AlarmDeleteAlarmStringID** attribute**.**

```
<AlarmConfiguration>
   <Alarm Name="Alarm_300.Value1.Lolo" ID="10"/>
   <Alarm Name="Alarm1*" ID="11"/>
</AlarmConfiguration>
```

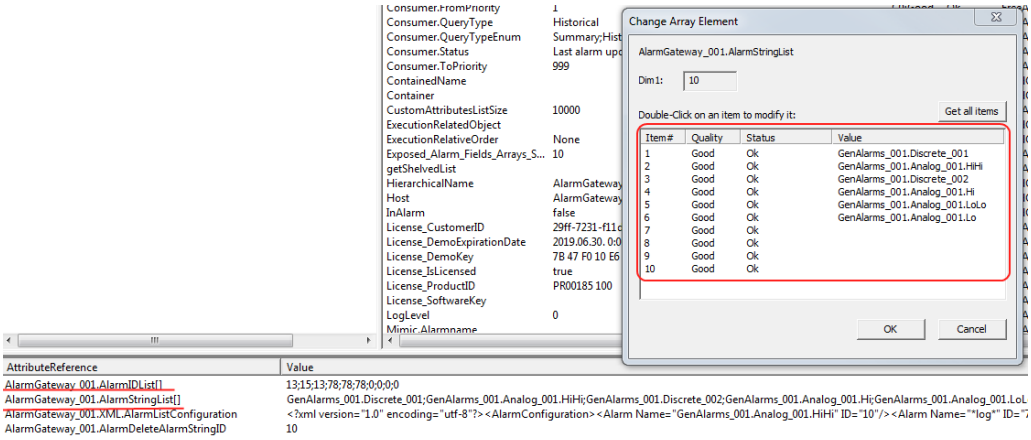
Note: For testing proposes in WAS following sample alarms are defined Analog_001 (LoLo,Lo,Hi,HiHi), Discreate_001, Discreate_002

e) set it to Alarm Gateway Big String **XML.AlarmListConfiguration** attribute.

f) When defined alarm raises



g) Exposed Alarm Field arrays are filled with defined Alarm Fields that are defined in **XML.AlarmListConfiguration** attribute:



Configuration attributes

XML.AlarmListConfiguration

Stores Alarm XML configuration

XML sample:

```
<AlarmConfiguration>
        <Alarm Name="*.HiHi" ID ="10"/>
</AlarmConfiguration>
```

Sample WAS script:

```
AlarmGateway_001.XML.AlarmListConfiguration =
"<AlarmConfiguration>" +
"<Alarm Name=""*.HiHi"" ID =""10""/>"+
"</AlarmConfiguration>";
```

AlarmIDList

dataType: int array

Array size: is defined in editor **Exposed_Alarm_Fields_Arrays_Size** attribute.

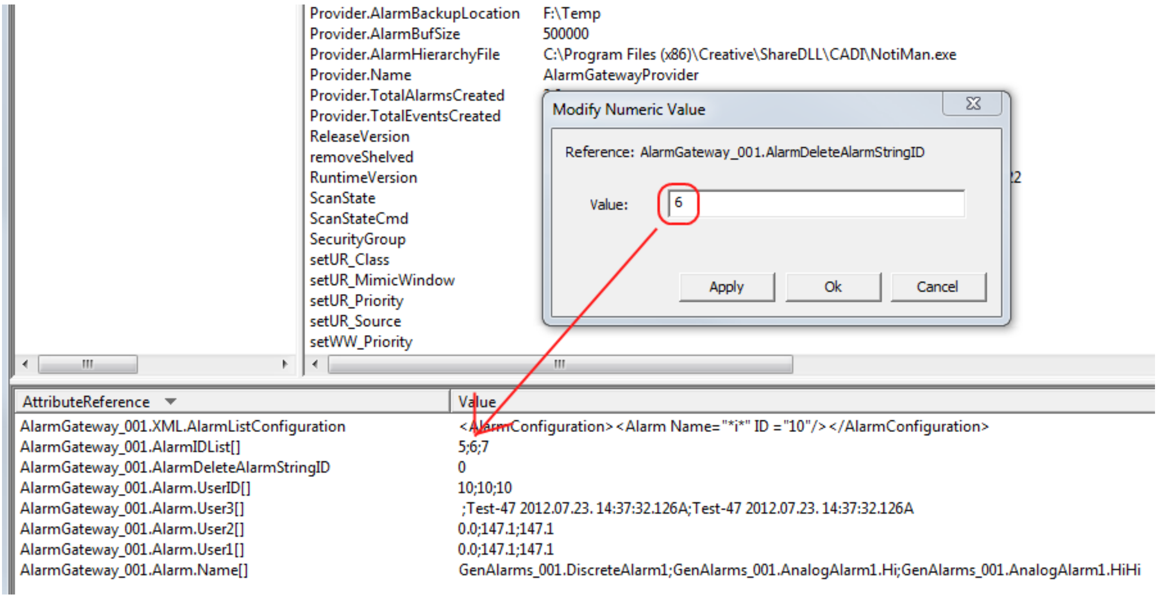
Description:
Array of Alarm Gateway generated exposed alarm unique ID that can be used to identify each exposed alarm.

AttributeReference	Value
AlarmGateway_001.XML.AlarmListConfiguration	<AlarmConfiguration><Alarm Name="*i*" ID ="10"/></AlarmConfiguration>
AlarmGateway_001.AlarmIDList[]	5;6;7
AlarmGateway_001.AlarmDeleteAlarmStringID	0
AlarmGateway_001.Alarm.UserID[]	10;10;10
AlarmGateway_001.Alarm.User3[]	;Test-47 2012.07.23. 14:37:32.126A;Test-47 2012.07.23. 14:37:32.126A
AlarmGateway_001.Alarm.User2[]	0.0;147.1;147.1
AlarmGateway_001.Alarm.User1[]	0.0;147.1;147.1
AlarmGateway_001.Alarm.Name[]	GenAlarms_001.DiscreteAlarm1;GenAlarms_001.AnalogAlarm1.Hi;GenAlarms_001.AnalogAlarm1.HiHi

AlarmDeleteAlarmStringID

dataType: int

Description:
Deletes entry from Exposed Alarm Fields that matches specified alarm ID in array **AlarmIDList.**



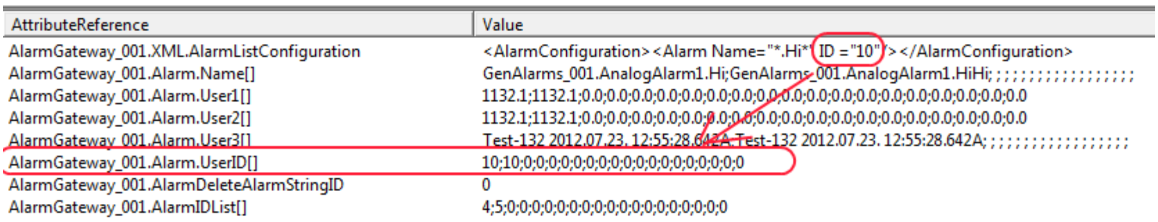
Alarm.UserID

dataType: int array

Description:

Displays User defined ID from Alarm list configuration (attribute XML.AlarmListConfiguration) file Alarm node attribute **ID**

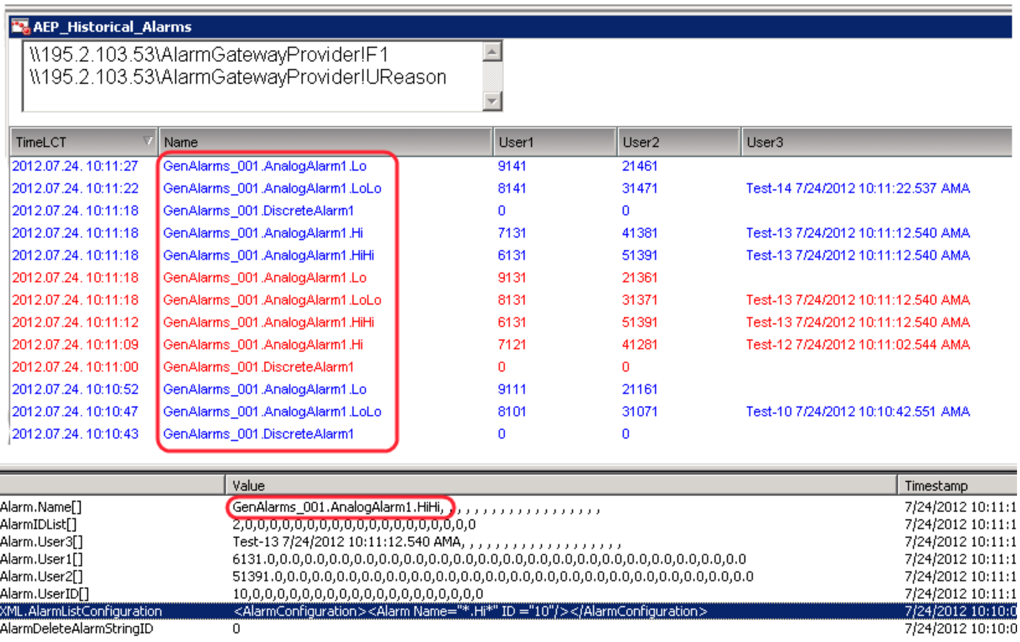
<AlarmConfiguration>
<Alarm Name="*.HiHi" **ID ="10"**/>
</AlarmConfiguration>



Exposed attributes

Alarm.Name

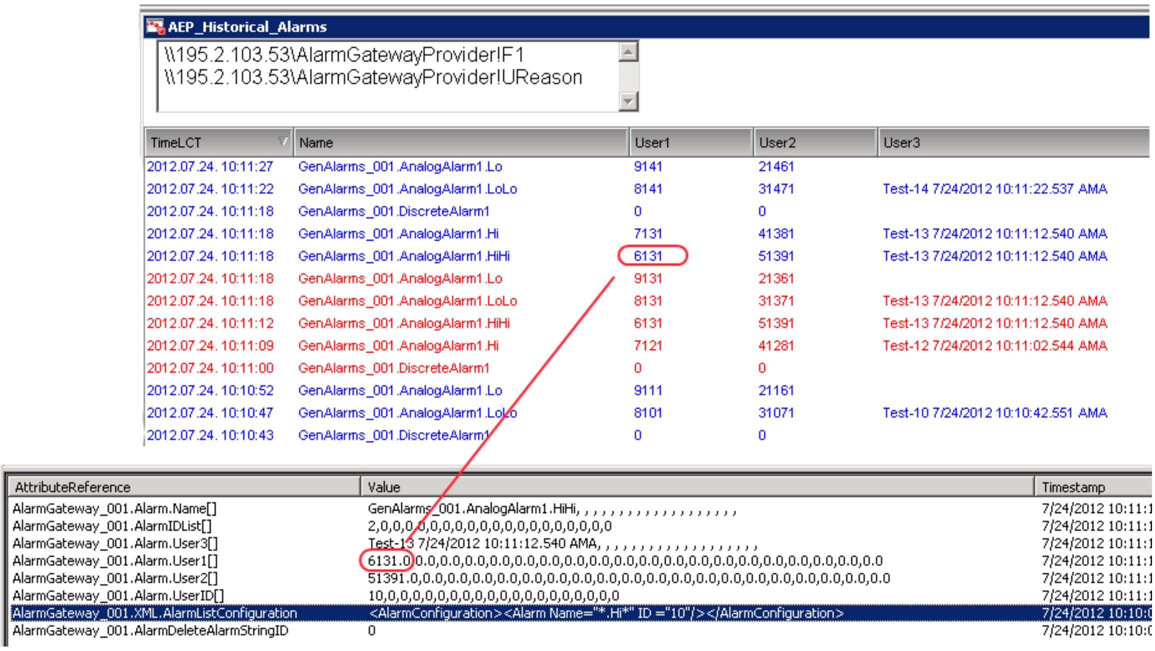
dataType: string array
Description: Exposed alarm name



Alarm.User1

dataType: string array
Array size: is defined in editor **Exposed_Alarm_Fields_Arrays_Size** attribute.

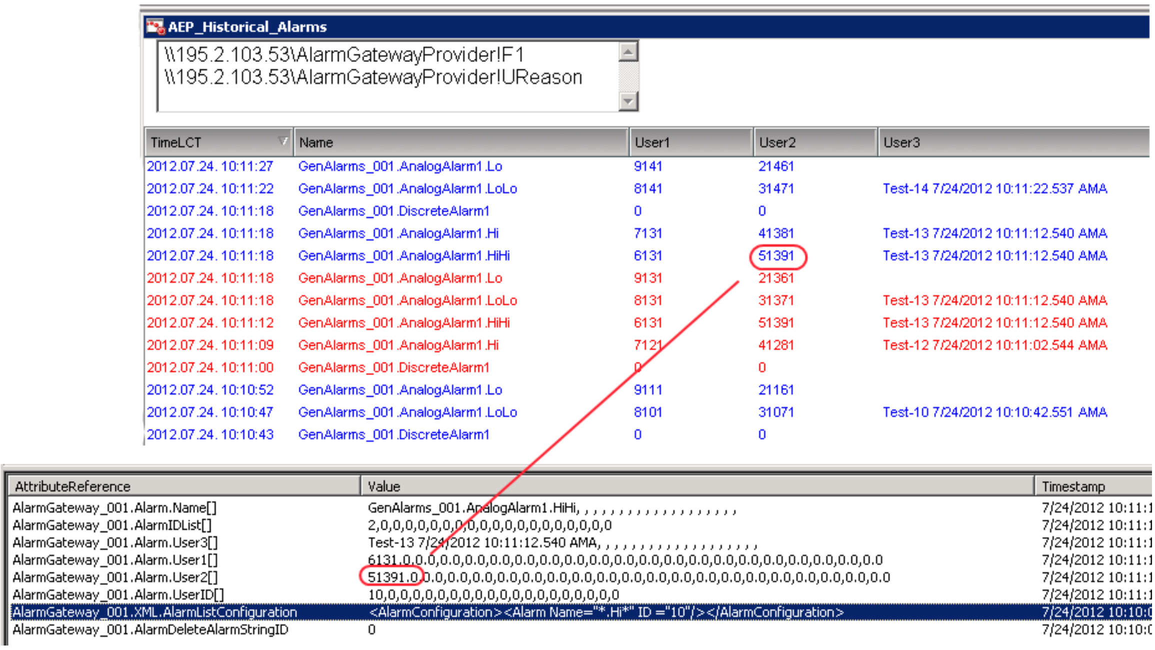
Description: Exposes Alarm field User1



Alarm.User2

dataType: float array
Array size: is defined in editor **Exposed_Alarm_Fields_Arrays_Size** attribute.

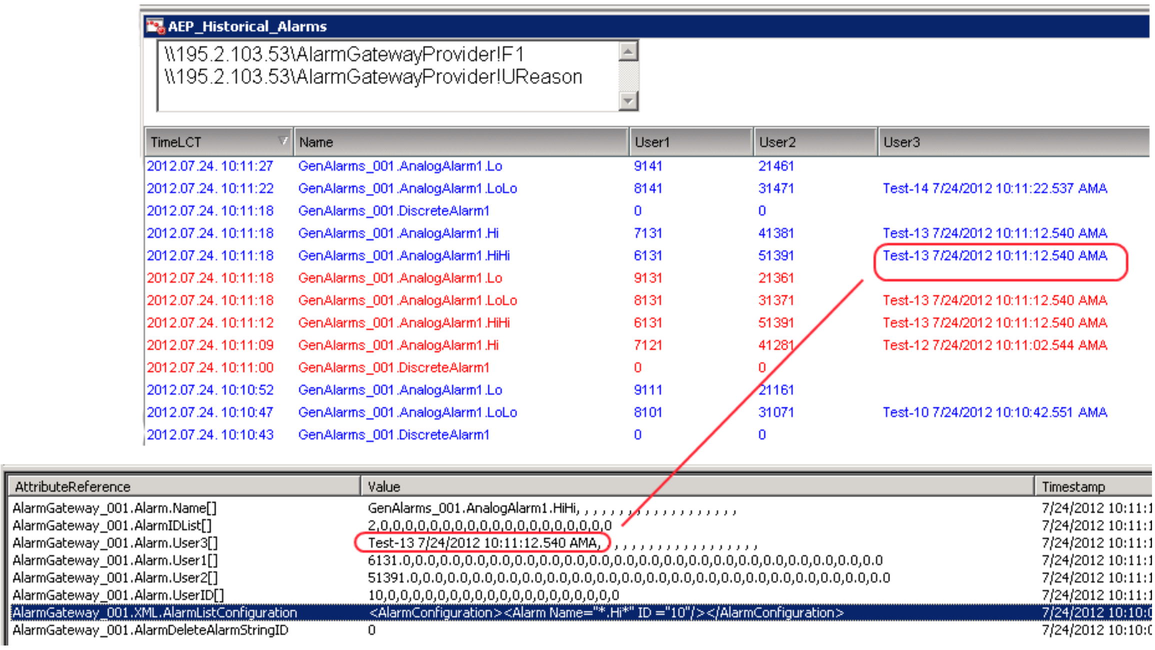
Description: Exposes Alarm field User2



Alarm.User3

dataType: float array
Array size: is defined in editor attribute **Exposed_Alarm_Fields_Arrays_Size** attribute.

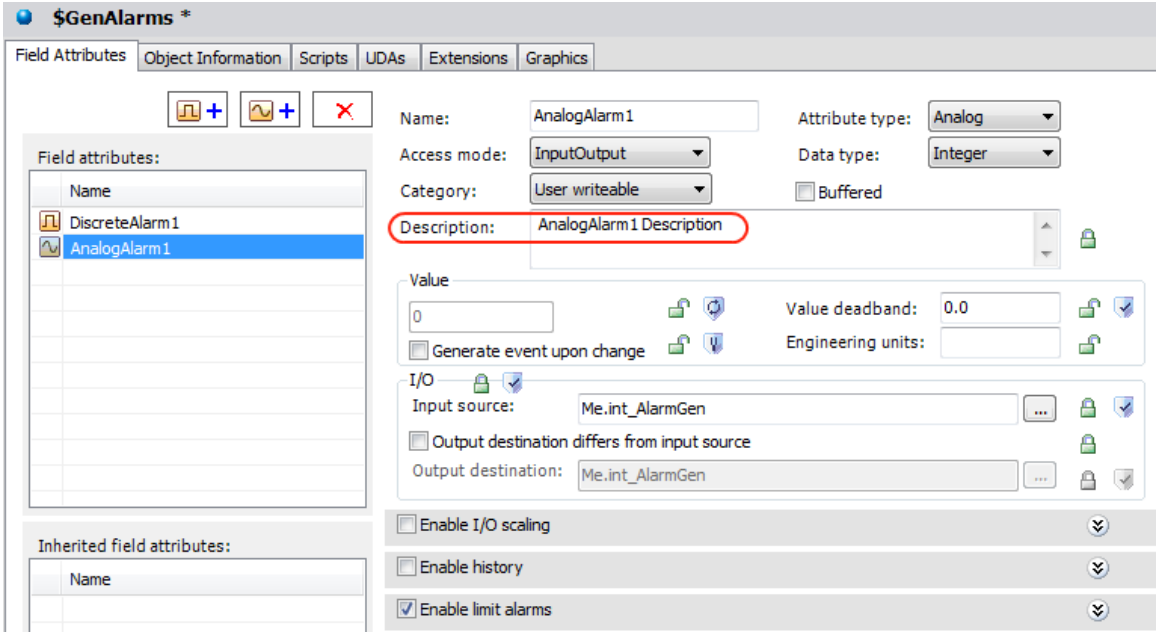
Description: Exposes Alarm field User3



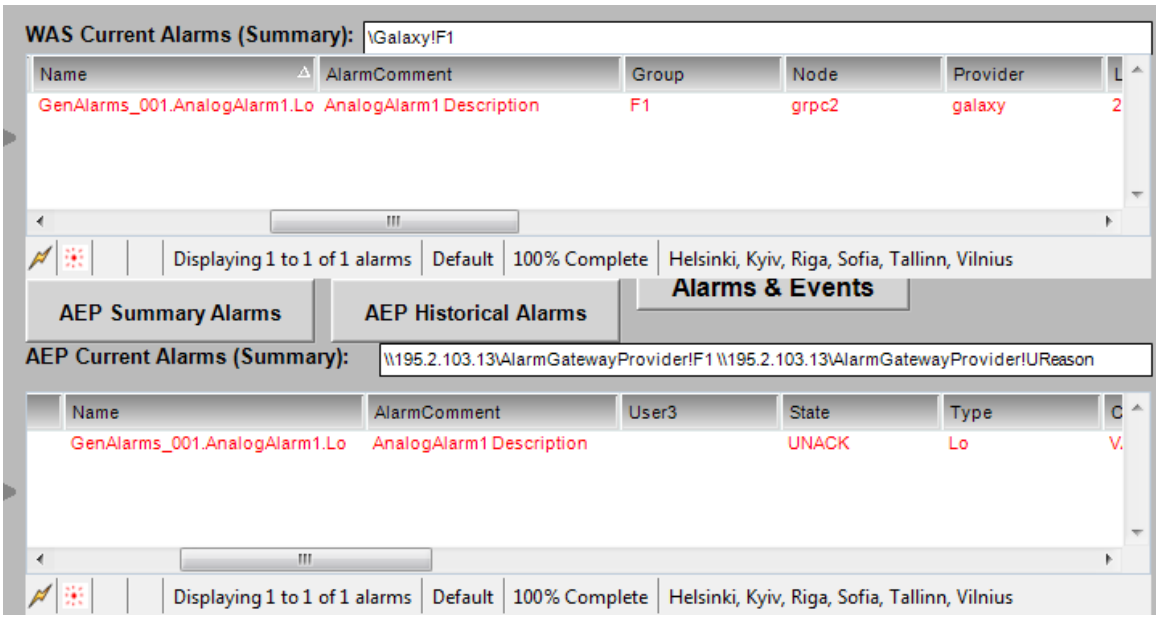
Custom Acked alarm comment field

Overview

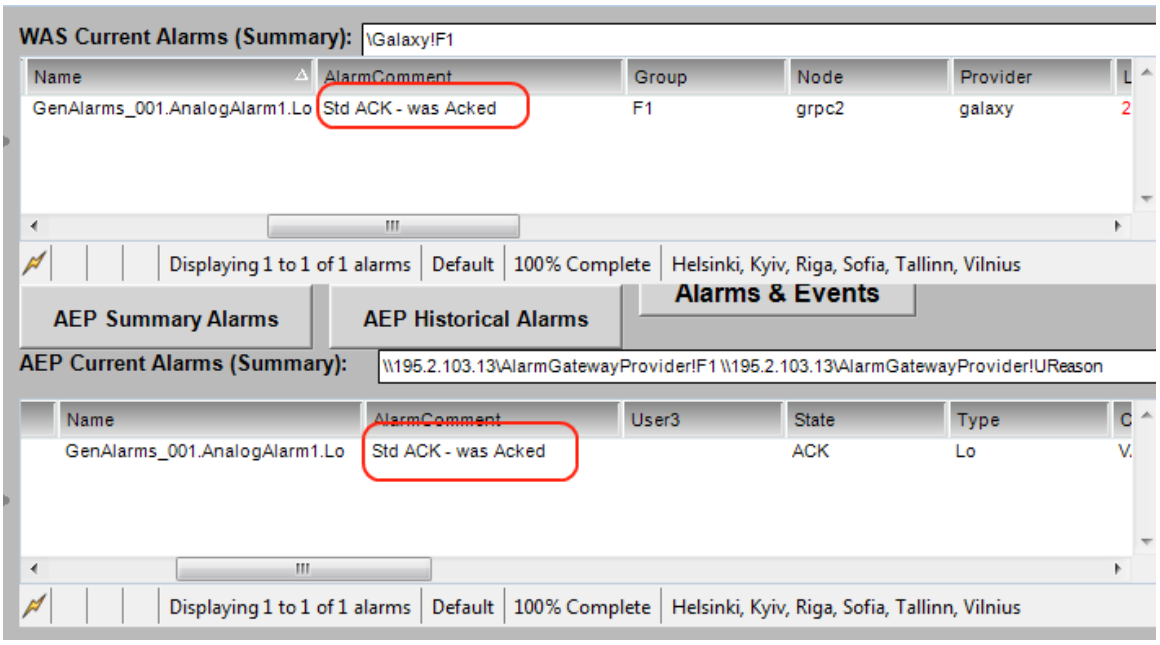
Using Alarm gateway is possible to configure custom Acked alarm comment field for Acked alarms this is useful if alarm descriptions are used:



Alarm description is stored in AlarmComment field:



If alarm is Acked by default AlarmComment value is overwritten by alarm comment value:

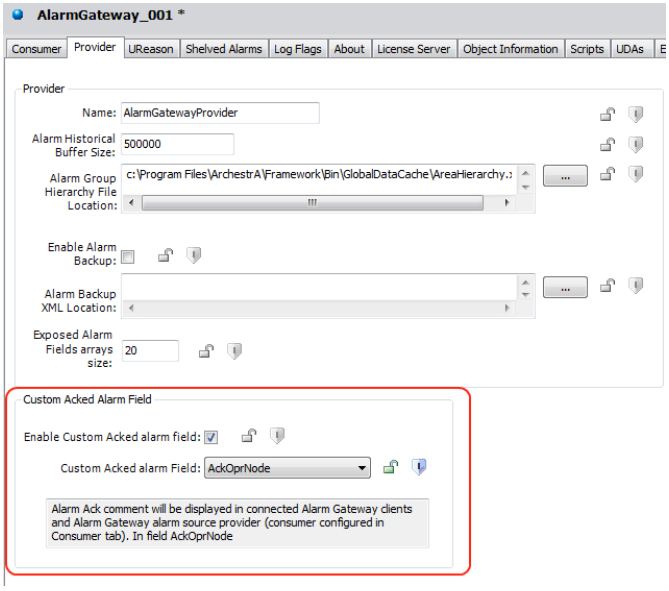


With Alarm Gateway is possible to configure different Alarm field to store Ack comment and preserve Alarm description.

Configuration

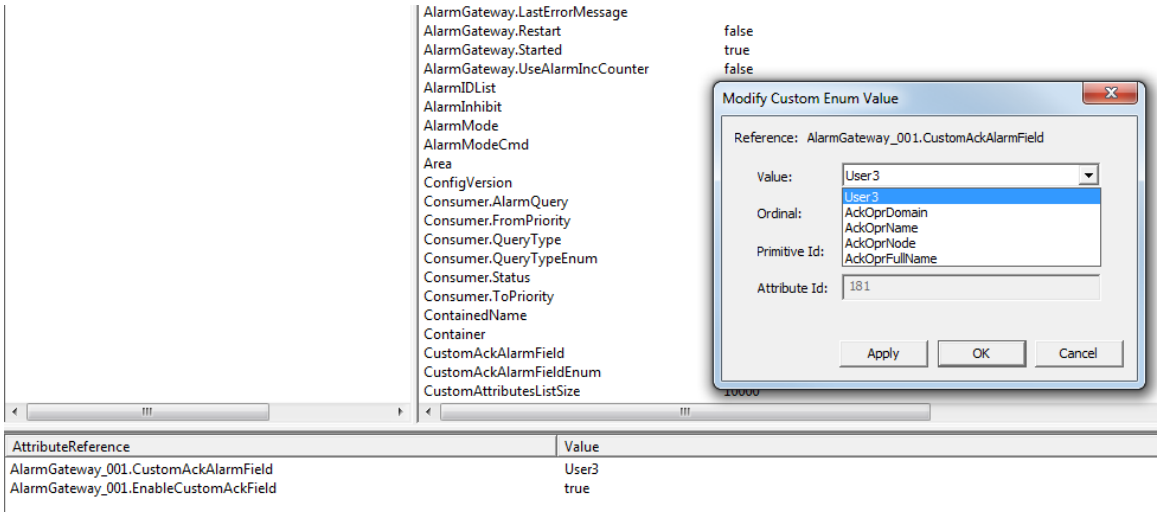
Editor

Custom alarm Ack comment field feature can be configured in tab **Provider** section **Custom Acked Alarm Field:**



Runtime

Custom alarm Acked comment field is possible to configure in runtime without Alarm gateway restart:



Attributes

EnableCustomAckField

Datatype: bool
Description: enables/disables Custom alarm Ack comment field feature.

If true - Custom alarm Ack comment field feature is enabled and Acked alarm comment is written in configured field in attribute CustomAckAlramField

If false - Custom alarm Ack comment field feature is disabled and default Alarm system functionality is used Ack comment overwrites field **AlarmComment**.

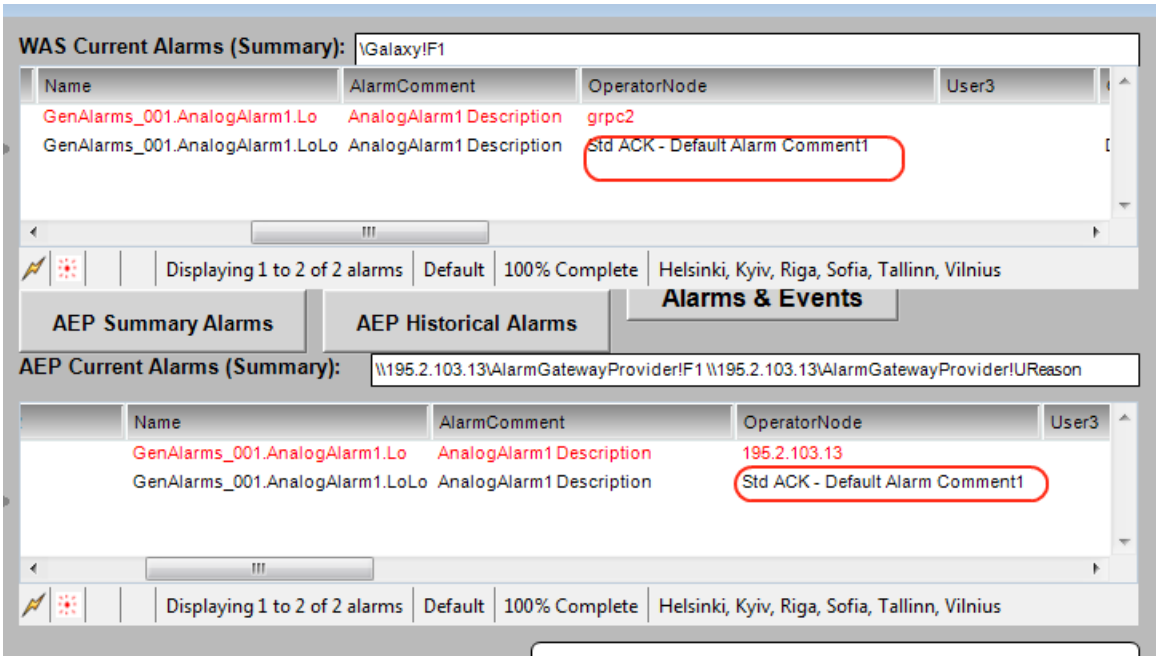
CustomAckAlramField

Datatype: Enum
Description: Defines custom Alarm Ack field.

Following custom Alarm Ack Comment field are possible to configure:

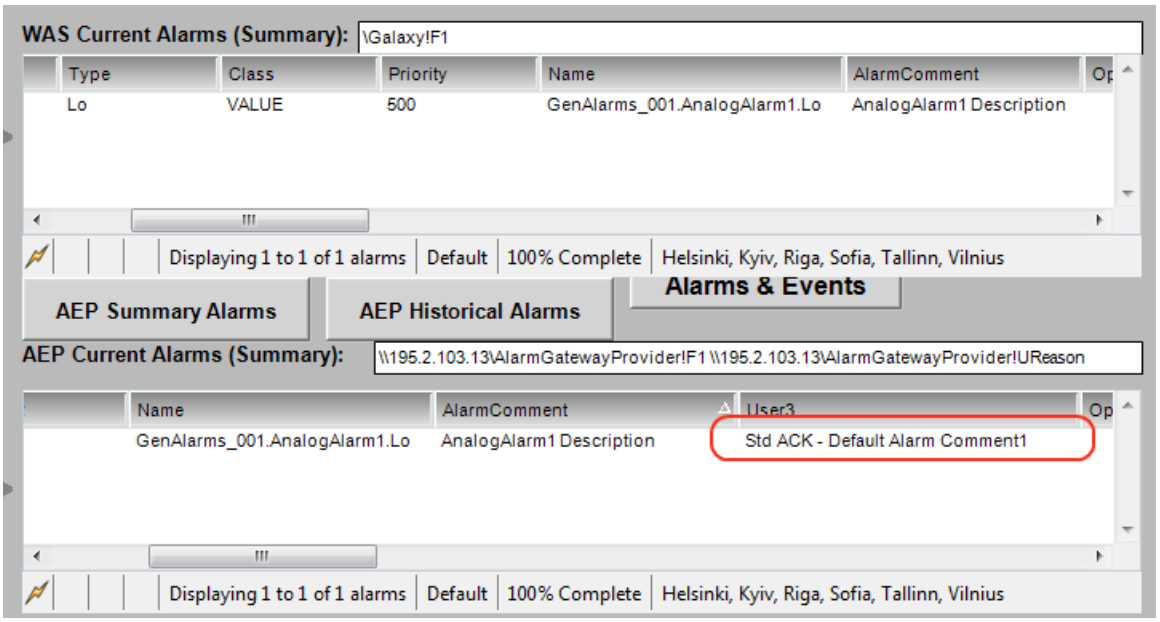
- **AckOprNode**

Alarm Ack comment will be displayed in connected Alarm Gateway clients and Alarm Gateway alarm source provider (consumer configured in Consumer tab). In field **AckOprNode**:



- **User3**

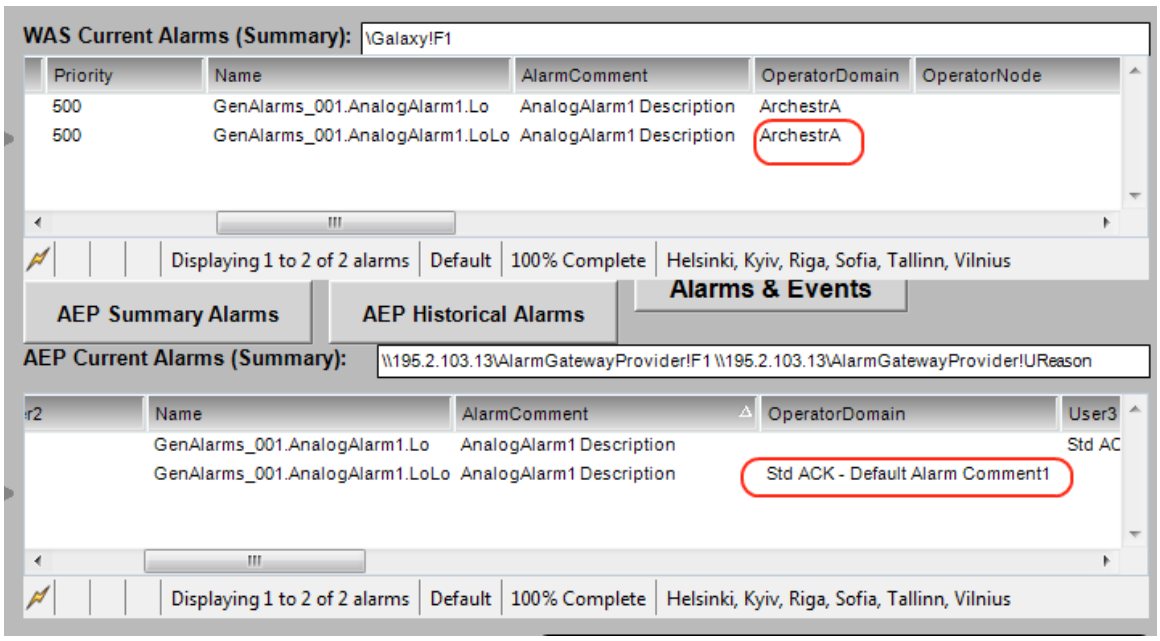
Alarm Ack comment will be displayed in connected Alarm Gateway clients only, in field **User3:**



Note: WAS alarm provider will only Ack alarm without changing AlarmComment.

- **AckOprDomain**

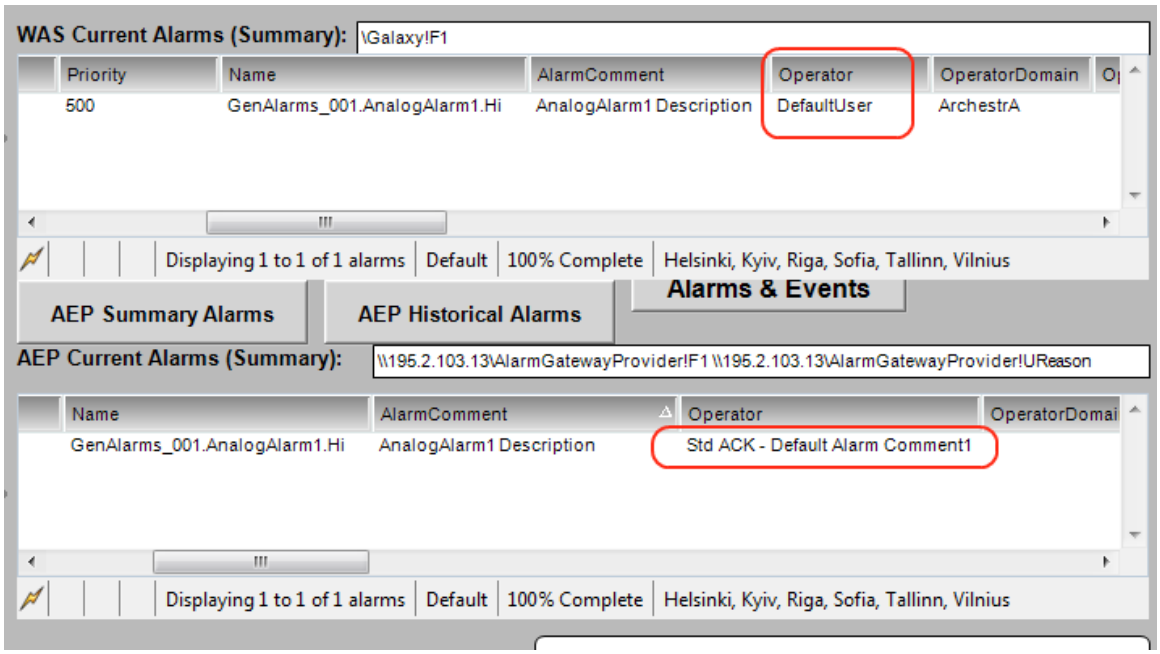
Alarm Ack comment will be displayed in connected Alarm Gateway clients only, in field **OperatorDomain:**



Note: WAS alarm provider will only Ack alarm without changing AlarmComment.

- **AckOprName**

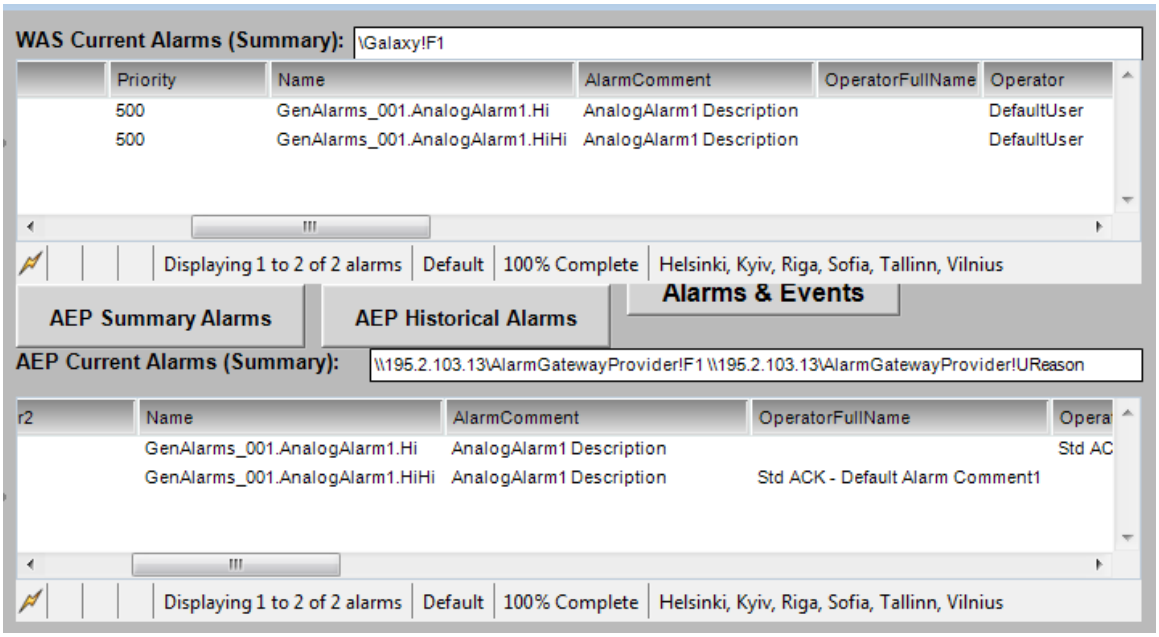
Alarm Ack comment will be displayed in connected Alarm Gateway clients only, in field **OperatorName:**



Note: WAS alarm provider will only Ack alarm without changing AlarmComment.

- **AckOprFullName**

Alarm Ack comment will be displayed in connected Alarm Gateway clients only, in field **OperatorFullName:**



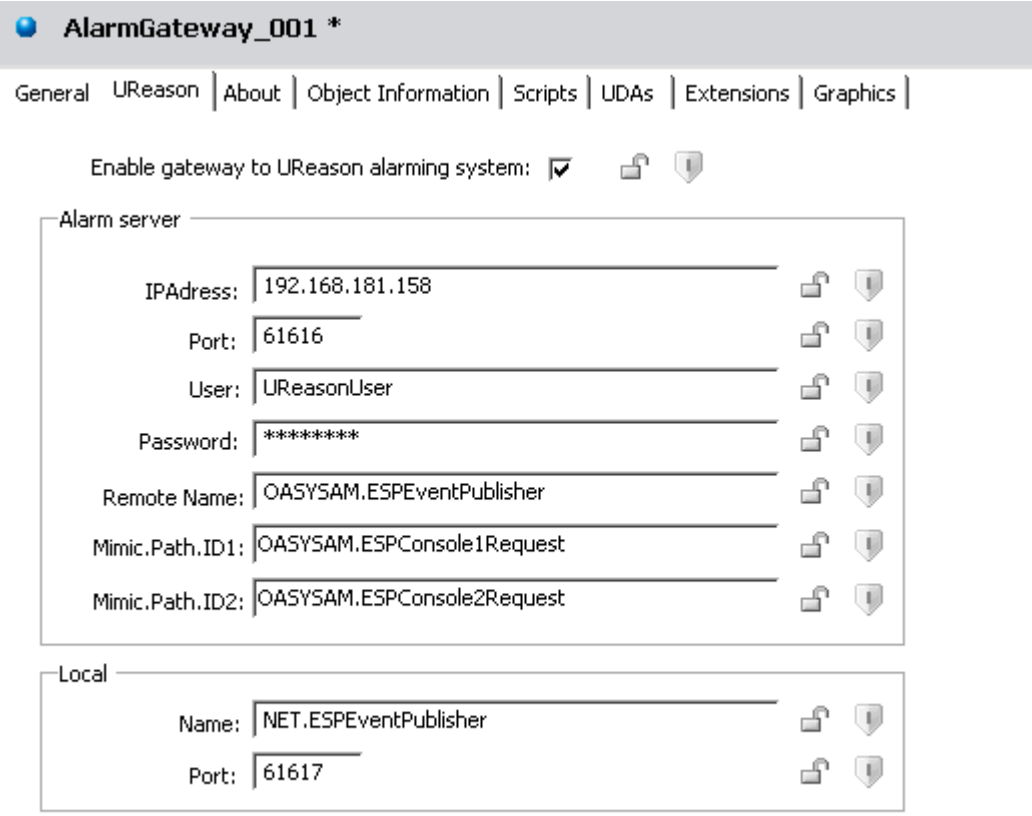
Note: WAS alarm provider will only Ack alarm without changing AlarmComment.

UReason gateway

Alarm Gateway Object provides functionality to send/receive alarms to/from Wonderware alarming system from/to UReason alarming system. The following functionality are supported:

1. Send new and acknowledged alarms to UReason alarm system.
2. UReason functionality to show Mimic InTouch windows.
3. UReason Shelved alarms functionality.
4. Acknowledge Wonderware alarms from UReason alarm system.

The following configuration is required for UReason gateway functionality:



Please, refer to UReason documentation for more information about UReason alarming system.

Alarm Gateway UReason Mimic functionality

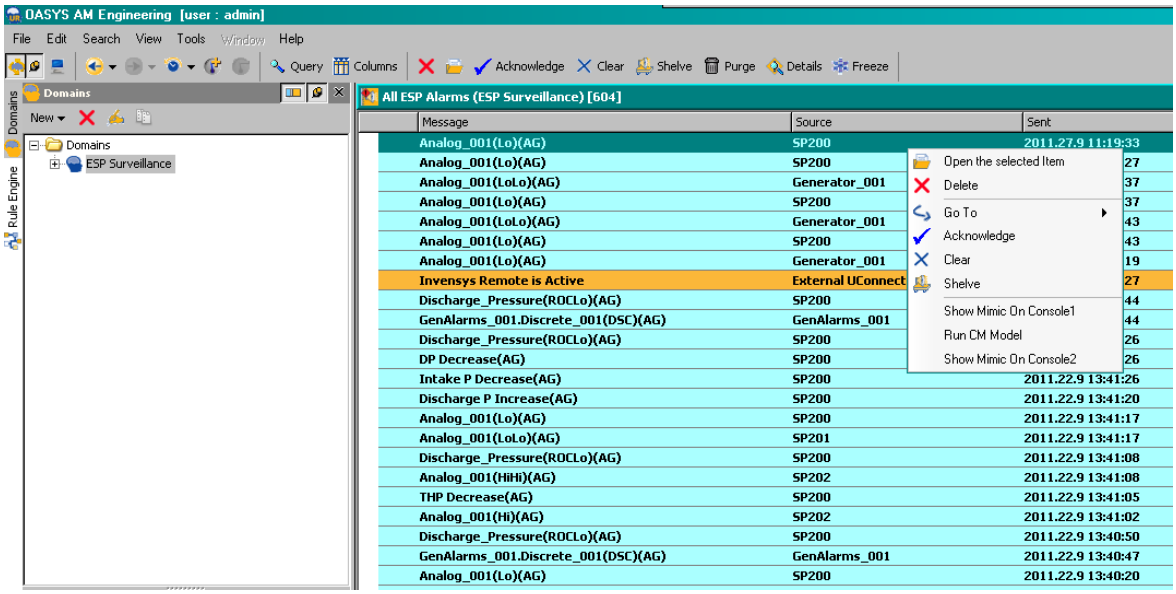
Alarm gateway supports UReason Mimic functionality.

Following object attributes are used for Console1 and Console2:

- AlarmGateway_001.UReason.Mimic.Path.Console1
  If user selects “Show Mimic On Console1” from UReason alarm menu, this attribute is changed to UReason Source value (for Alarm Analog_001.Lo it is SP200, see picture below).

- AlarmGateway_001.UReason.Mimic.Path.Console2

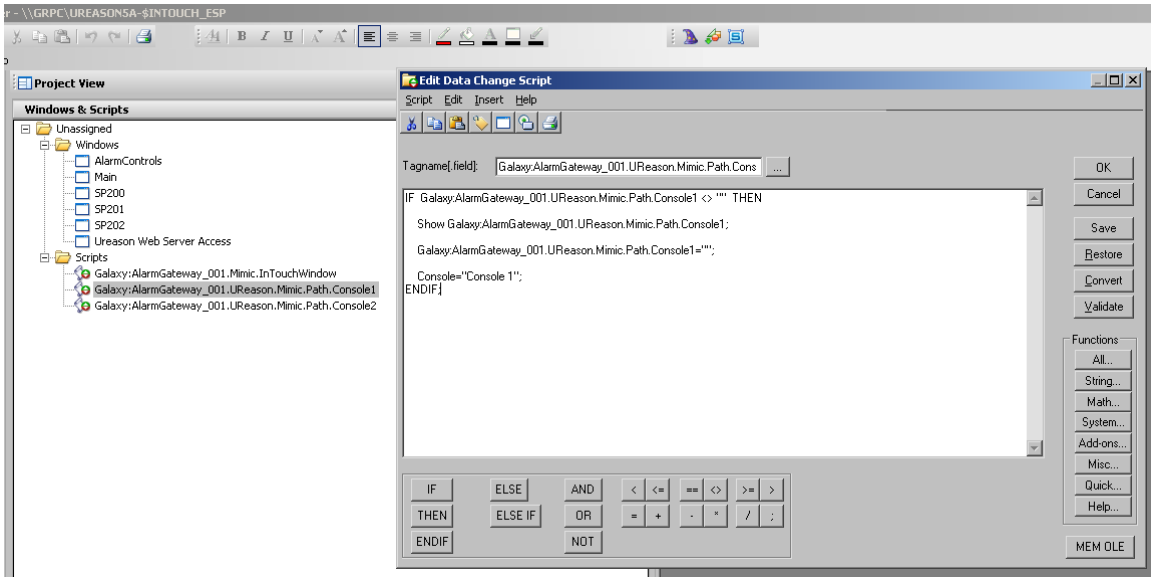
If user selects “Show Mimic On Console2” from UReason alarm menu, this attribute is changed to UReason Source value (for Alarm Analog_001.Lo it is SP200 see picture below).



Mimic functionality can be used for opening specific InTouch windows that are tied to UReason alarm by the Source value.

Sample InTouch script:

```
IF  Galaxy:AlarmGateway_001.UReason.Mimic.Path.Console1 <> ""  THEN

    Show Galaxy:AlarmGateway_001.UReason.Mimic.Path.Console1;

    Galaxy:AlarmGateway_001.UReason.Mimic.Path.Console1="";

    Console="Console 1";
ENDIF;


IF  Galaxy:AlarmGateway_001.UReason.Mimic.Path.Console2 <> ""  THEN

    Show Galaxy:AlarmGateway_001.UReason.Mimic.Path.Console2;

    Galaxy:AlarmGateway_001.UReason.Mimic.Path.Console2="";

    Console="Console 2";
ENDIF;
```



Mimic functionally without UReason alarming system

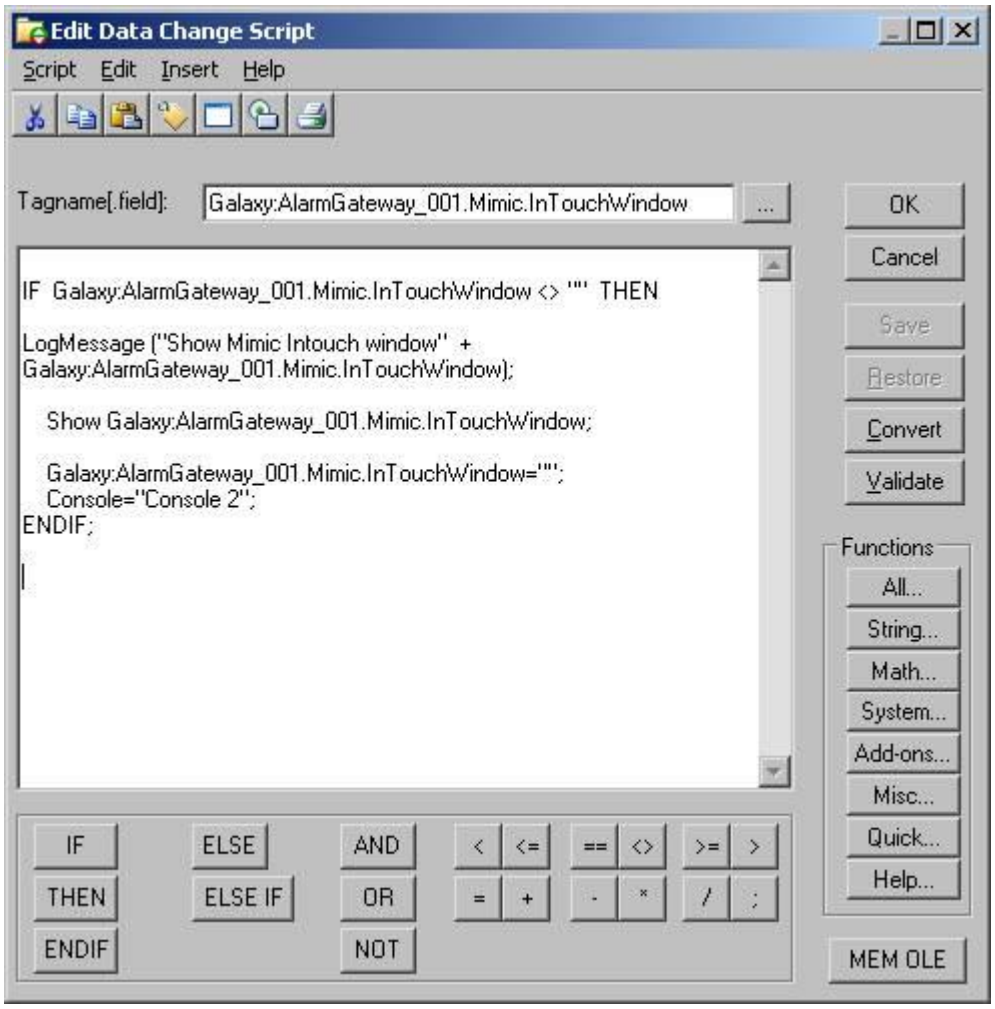
For Alarm Gateway UReason Mimic functionality only for Wonderware alarm system (without UReason), the following string attributes are needed:

- **AlarmGateway_001.Mimic.Alarmname** – input AlarmName from provider alarms list (max length 32 characters)
  Sample: SP200.Intake_Pressure_Decrease

- **AlarmGateway_001.Mimic.InTouchWindow** - returns default (WAS object name) or user-defined (set in **setSource** attribute) value, e.g. SP200

<u>Sample script:</u>

```
InTouch data change script Galaxy:AlarmGateway_001.Mimic.InTouchWindow

IF  Galaxy:AlarmGateway_001.Mimic.InTouchWindow <> ""  THEN

LogMessage ("Show Mimic Intouch window"  +
Galaxy:AlarmGateway_001.Mimic.InTouchWindow);

   Show Galaxy:AlarmGateway_001.Mimic.InTouchWindow;

   Galaxy:AlarmGateway_001.Mimic.InTouchWindow="";
   Console="Console 2";
ENDIF;
```


Edit Data Change Script
Script Edit Insert Help
Tagname[.field]:
Galaxy:AlarmGateway_001.Mimic.InTouchWindow
IF Galaxy:AlarmGateway_001.Mimic.InTouchWindow <> "" THEN
LogMessage ("Show Mimic Intouch window" +
Galaxy:AlarmGateway_001.Mimic.InTouchWindow);
Show Galaxy:AlarmGateway_001.Mimic.InTouchWindow;
Galaxy:AlarmGateway_001.Mimic.InTouchWindow="";
Console="Console 2";
ENDIF;
OK
Cancel
Save
Restore
Convert
Validate
Functions
All...
String...
Math...
System...
Add-ons...
Misc...
Quick...
Help...
IF
THEN
ENDIF
ELSE
ELSE IF
AND
OR
NOT
MEM OLE

Multiple distributed Alarm Gateway configuration setup sample

In high load WAS solutions were total alarm exceeds several hundreds or thousands and additional high CPU load features like **Wonderware alarm system custom attributes** are used is highly recommended to use several Alarm Gateway object instances to avoid object overload.

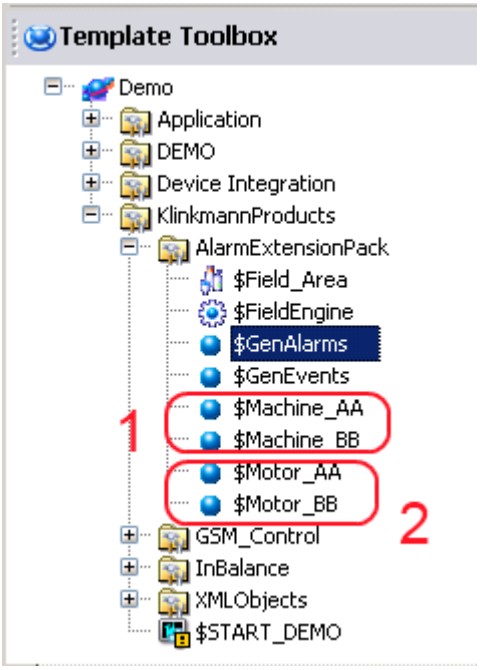
Following sample is for two Alarm Gateway instances.

Object configuration

1) Define template/instance sets for each Alarm Gateway instance, lets assume we are planning to use 2 Alarm Gateway instances (**AlarmGateway_001** and **AlarmGateway_002**):

   a) For **AlarmGateway_001**
      Define templates: $**Machine_AA** and $**Machine_BB**

   b) For **AlarmGateway_002**
      Define templates: $**Motor_AA** and $ **Motor_BB**



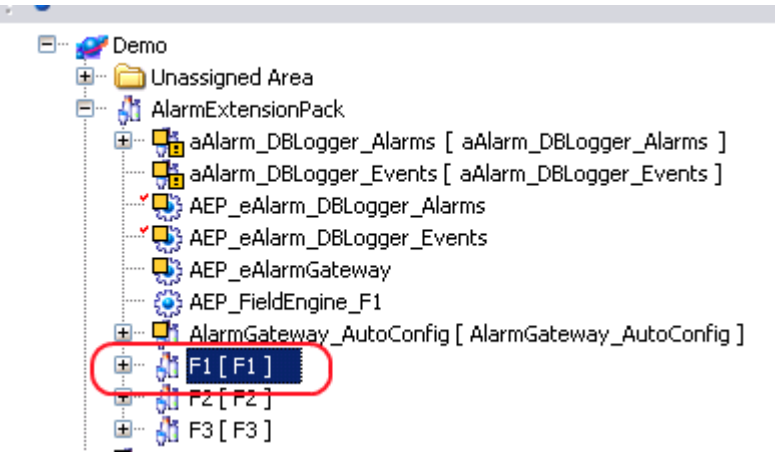
2) Edit template ($**Machine_AA** and $**Machine_BB**) scripts to point to assigned object **AlarmGateway_001**

```
' test 2 set new  priority each 10 sec
if(Me.SimCustomAttributes == true) THEN

if(Me.iCounter > 999) THEN
      Me.iCounter = 1; ' reset
ELSE
      Me.iCounter = Me.iCounter + 1;
ENDIF;
' User1
      AlarmGateway_001.setWW_User1 =  Me.Tagname + ".AnalogAlarm1.Lo=1"+ Me.iCounter+ ",1";
      AlarmGateway_001.setWW_User1 =  Me.Tagname + ".AnalogAlarm1.LoLo=1"+ Me.iCounter+ ",1";
      AlarmGateway_001.setWW_User1 =  Me.Tagname + ".AnalogAlarm1.Hi=1"+ Me.iCounter+ ",1";
      AlarmGateway_001.setWW_User1 =  Me.Tagname + ".AnalogAlarm1.HiHi=1"+ Me.iCounter+ ",1";

' User2
      AlarmGateway_001.setWW_User2 =  Me.Tagname + ".AnalogAlarm1.Lo=1"+ Me.iCounter+ ",1";
      AlarmGateway_001.setWW_User2 =  Me.Tagname + ".AnalogAlarm1.LoLo=1"+ Me.iCounter+ ",1";
      AlarmGateway_001.setWW_User2 =  Me.Tagname + ".AnalogAlarm1.Hi=1"+ Me.iCounter+ ",1";
      AlarmGateway_001.setWW_User2 =  Me.Tagname + ".AnalogAlarm1.HiHi=1"+ Me.iCounter+ ",1";
' User3
      AlarmGateway_001.setWW_User3 =  Me.Tagname +
 ".Analog_001.Lo=Test-"+ Me.iCounter+ " "+ System.DateTime.Now + "A";
      AlarmGateway_001.setWW_User3 =  Me.Tagname + ".AnalogAlarm1.LoLo=Test-"+ Me.iCounter+ " "
+ System.DateTime.Now + "A";
      AlarmGateway_001.setWW_User3 =  Me.Tagname + ".AnalogAlarm1.Hi=Test-"+ Me.iCounter+ " "+
 System.DateTime.Now + "A";
      AlarmGateway_001.setWW_User3 =  Me.Tagname + ".AnalogAlarm1.HiHi=Test-"+ Me.iCounter+ " "
+ System.DateTime.Now + "A";
```

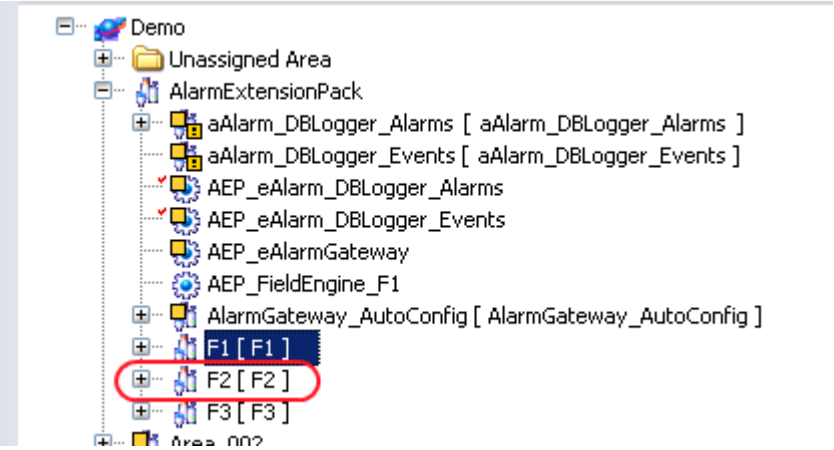
3) Edit template ($**Motor_AA** and $**Motor_BB**) scripts to point to assigned object **AlarmGateway_002**

```
' test 2 set new  priority each 10 sec
if(Me.SimCustomAttributes == true) THEN

if(Me.iCounter > 999) THEN
      Me.iCounter = 1; ' reset
ELSE
      Me.iCounter = Me.iCounter + 1;
ENDIF;
' User1
      AlarmGateway_002.setWW_User1 =  Me.Tagname + ".AnalogAlarm1.Lo=1"+ Me.iCounter+ ",1";
      AlarmGateway_002.setWW_User1 =  Me.Tagname + ".AnalogAlarm1.LoLo=1"+ Me.iCounter+ ",1";
      AlarmGateway_002.setWW_User1 =  Me.Tagname + ".AnalogAlarm1.Hi=1"+ Me.iCounter+ ",1";
      AlarmGateway_002.setWW_User1 =  Me.Tagname + ".AnalogAlarm1.HiHi=1"+ Me.iCounter+ ",1";

' User2
      AlarmGateway_002.setWW_User2 =  Me.Tagname + ".AnalogAlarm1.Lo=1"+ Me.iCounter+ ",1";
      AlarmGateway_002.setWW_User2 =  Me.Tagname + ".AnalogAlarm1.LoLo=1"+ Me.iCounter+ ",1";
      AlarmGateway_002.setWW_User2 =  Me.Tagname + ".AnalogAlarm1.Hi=1"+ Me.iCounter+ ",1";
      AlarmGateway_002.setWW_User2 =  Me.Tagname + ".AnalogAlarm1.HiHi=1"+ Me.iCounter+ ",1";
' User3
      AlarmGateway_002.setWW_User3 =  Me.Tagname +
 ".Analog_001.Lo=Test-"+ Me.iCounter+ " "+ System.DateTime.Now + "A";
      AlarmGateway_002.setWW_User3 =  Me.Tagname + ".AnalogAlarm1.LoLo=Test-"+ Me.iCounter+ " "
+ System.DateTime.Now + "A";
      AlarmGateway_002.setWW_User3 =  Me.Tagname + ".AnalogAlarm1.Hi=Test-"+ Me.iCounter+ " "+
 System.DateTime.Now + "A";
      AlarmGateway_002.setWW_User3 =  Me.Tagname + ".AnalogAlarm1.HiHi=Test-"+ Me.iCounter+ " "
+ System.DateTime.Now + "A";
```

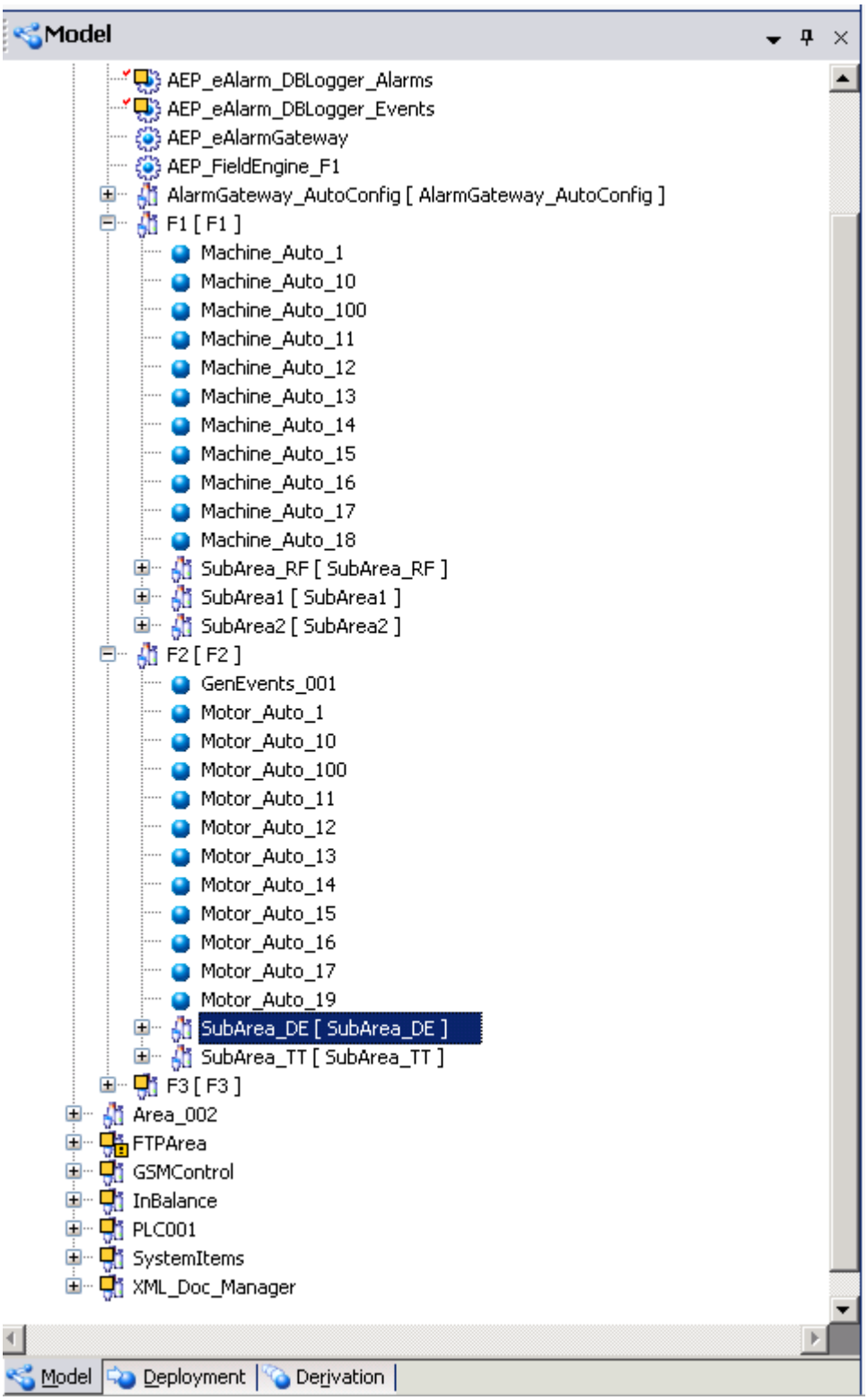
4) Create Area object **F1** area that will be used to all first set objects ($**Machine_AA** and $**Machine_BB**)



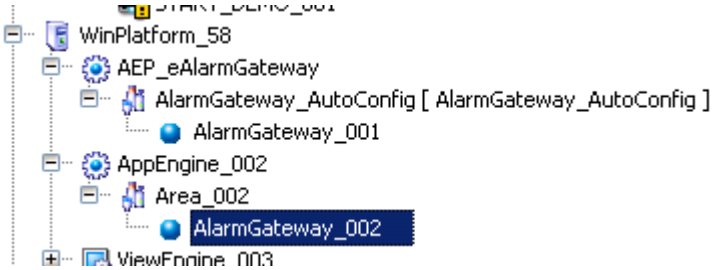
5) Create Area object **F2** area that will be used to all second set objects ($**Motor_AA** and $ **Motor_BB**



6) Create instances and deploy for both sets. Sample:

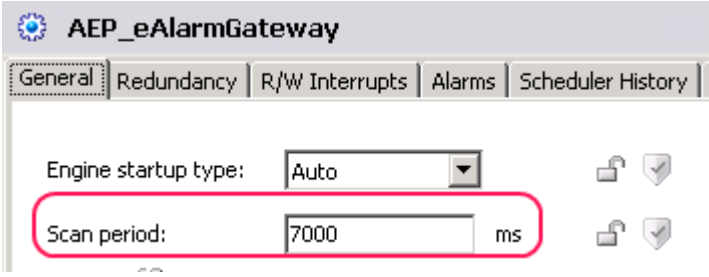


7) Create separate engines and Areas for each Alarm Gateway object instance:



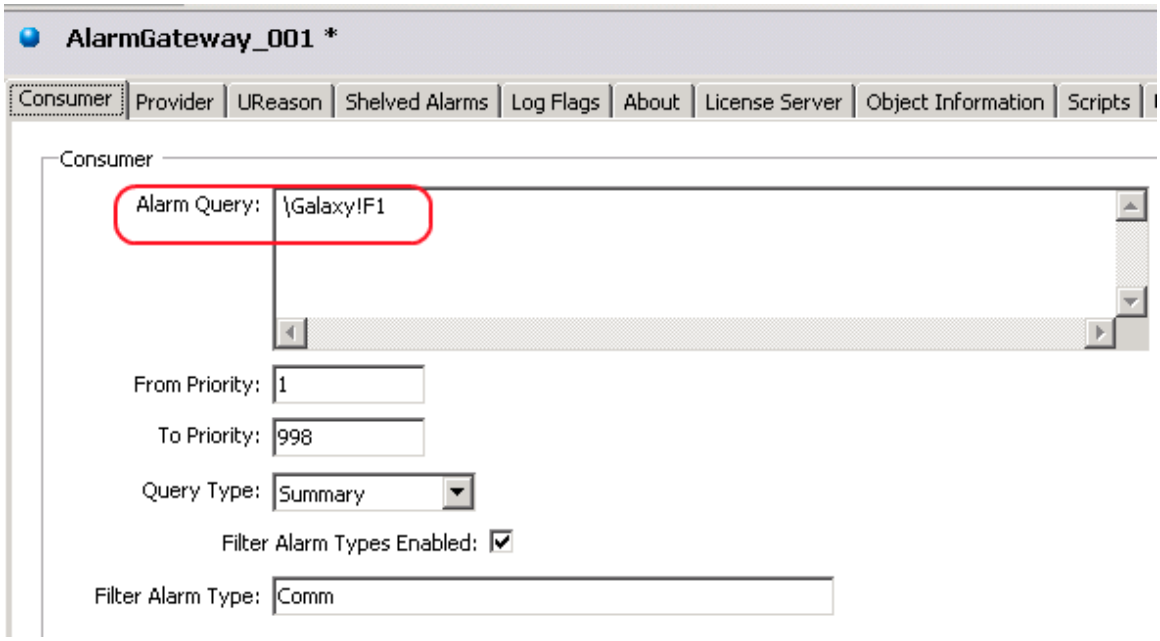
Note: Separate engine for each Alarm Gateway is **must have** requirement since each engine creates a separate process and that allows to distribute load between multiple CPU cores.

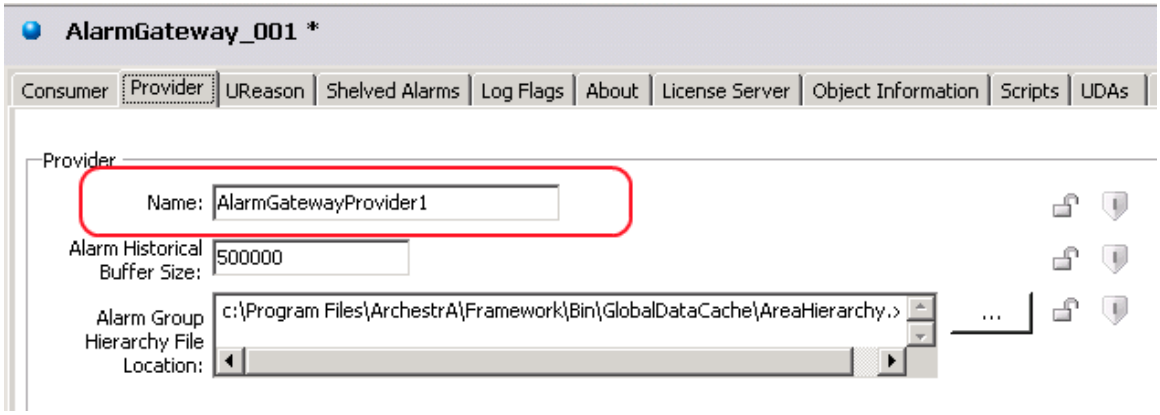
8) Set both engines Scan period to 7000 ms.



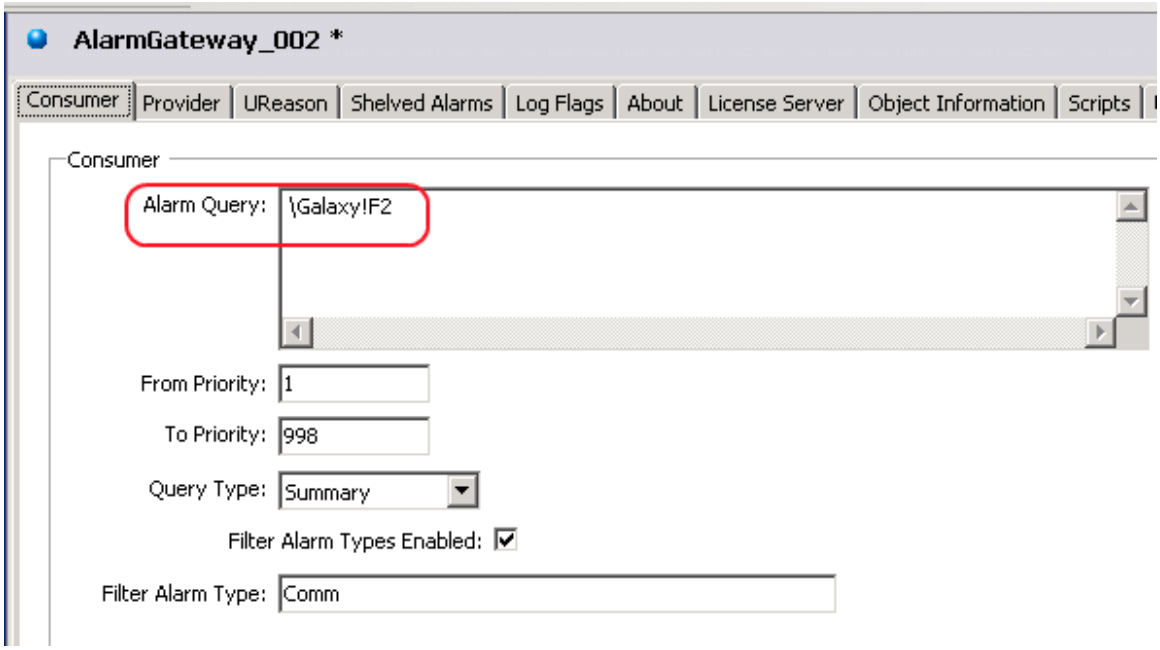
9) Configure both Instances of Alarm Gateway:

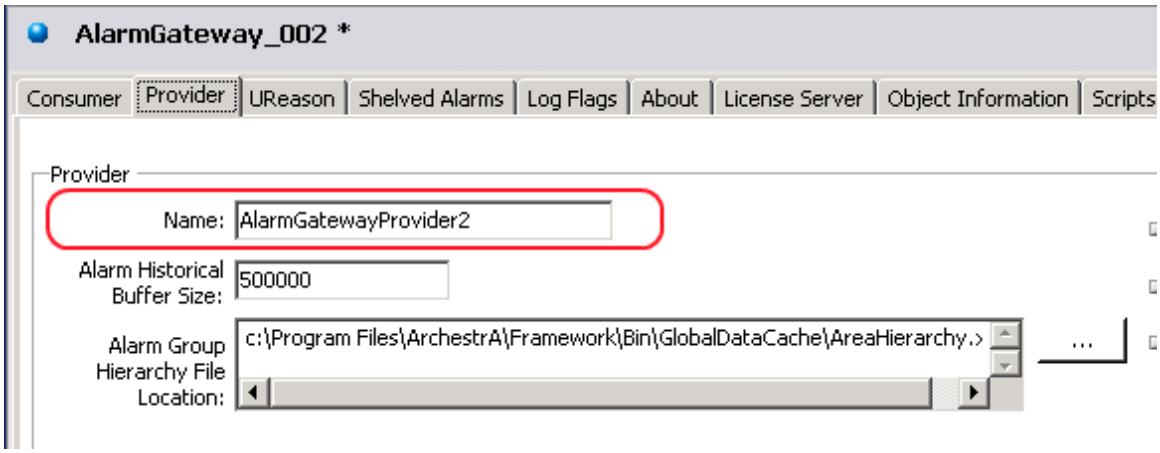
a. AlarmGateway_001
Alarm Query: **\Galaxy!F1**
Provider name: **AlarmGatewayProvider1**



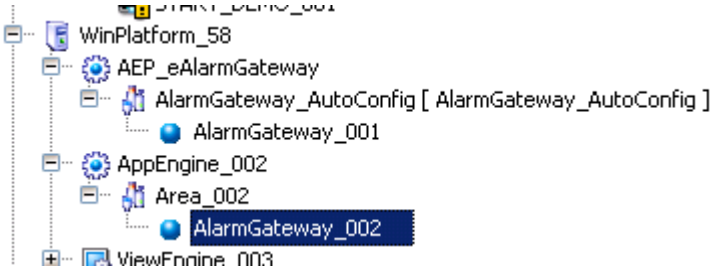


a. AlarmGateway_002
   Alarm Query: **\Galaxy!F2**
   Provider name: **AlarmGatewayProvider2**

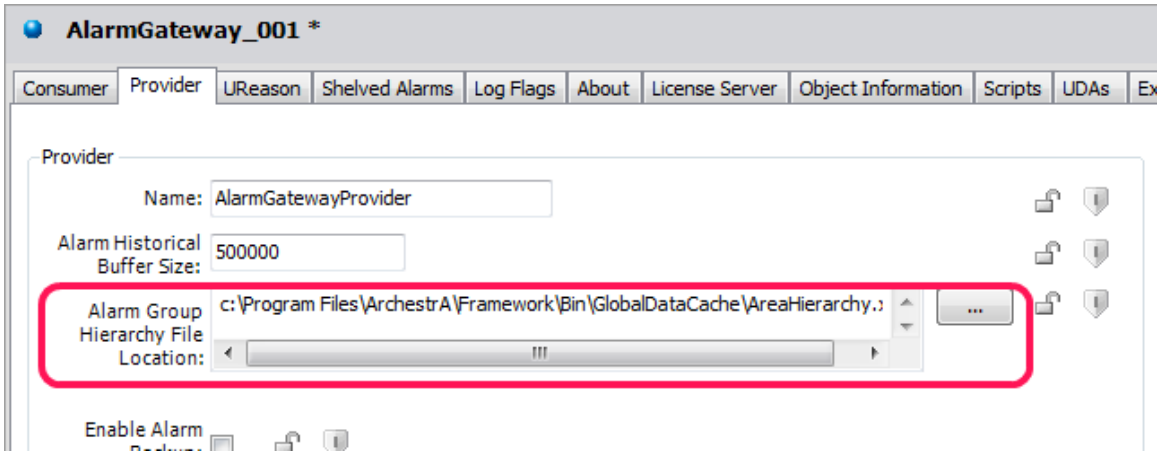




10) Deploy both instances of alarm Gateway



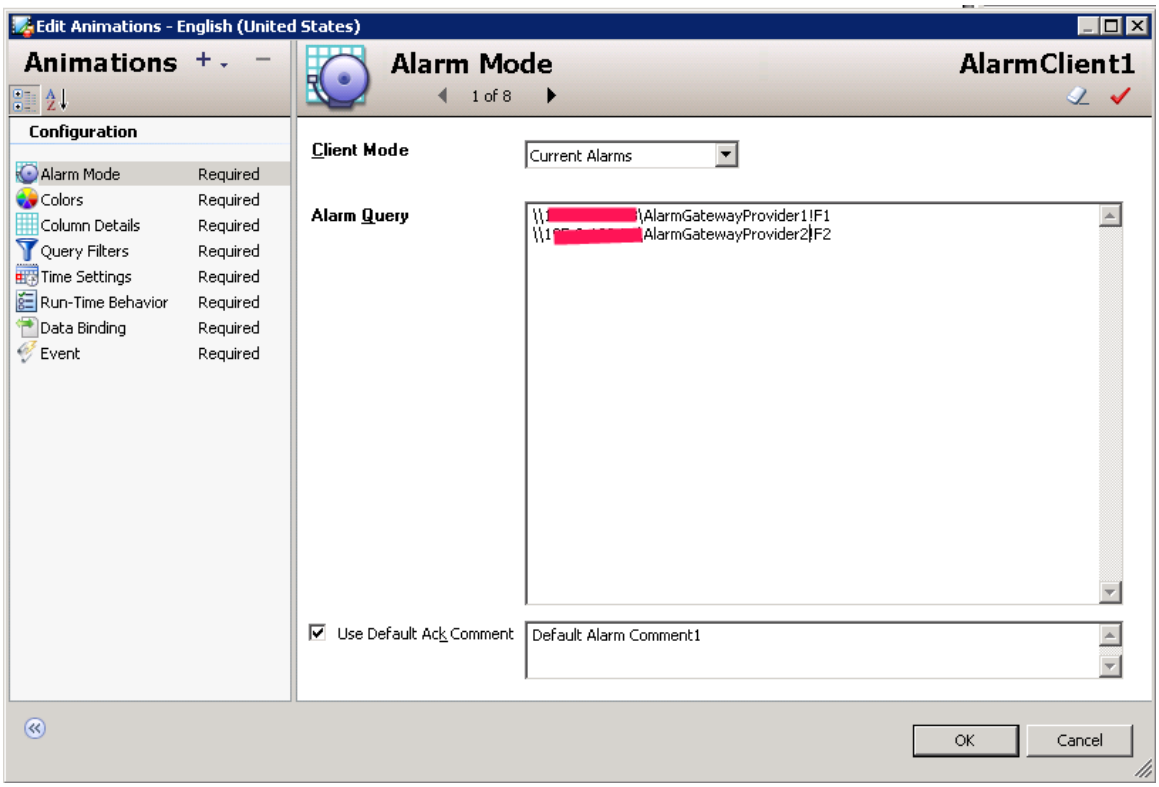
Note: if Area hierarchy is changed (new area is added or existing is moved) is required to redeploy AlarmGateway instance to update **Alarm Group Hierarchy File** information.

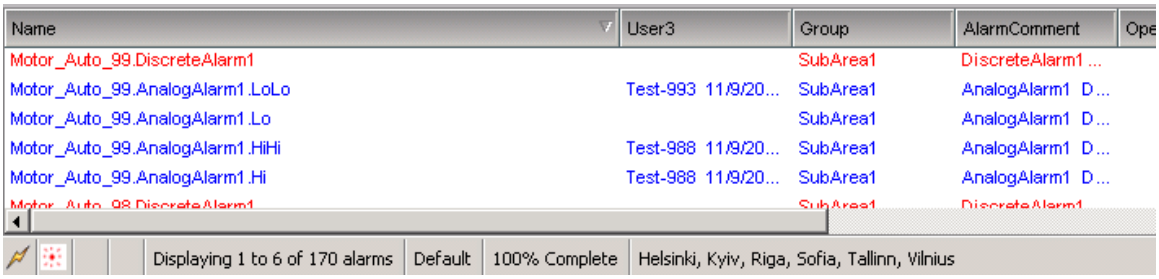


Alarm client configuration

To access of both AlarmGateway instances set following Alarm query in AlarmClient:

\\Hostname\AlarmGatewayProvider1!F1 \\Hostname\AlarmGatewayProvider2!F2


Edit Animations - English (United States)
Animations
Alarm Mode
1 of 8
AlarmClient1
Configuration
Alarm Mode Required
Colors Required
Column Details Required
Query Filters Required
Time Settings Required
Run-Time Behavior Required
Data Binding Required
Event Required
Client Mode
Current Alarms
Alarm Query
\\1[redacted]\AlarmGatewayProvider1!F1
\\1[redacted]\AlarmGatewayProvider2!F2
Use Default Ack Comment
Default Alarm Comment1
OK
Cancel

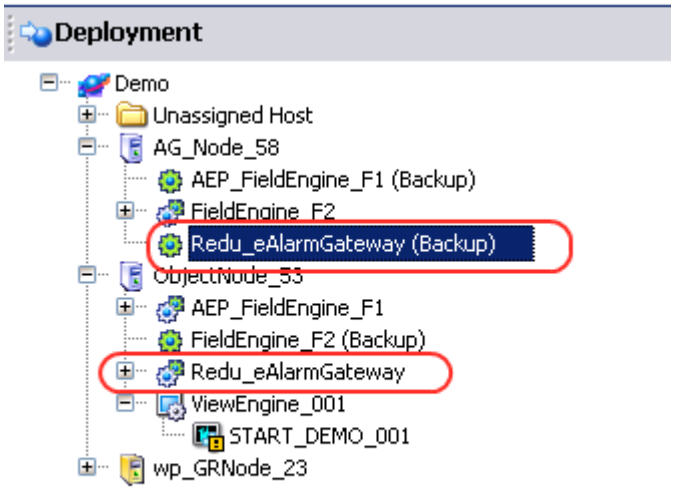

Name
User3
Group
AlarmComment
Motor_Auto_99.DiscreteAlarm1 SubArea1 DiscreteAlarm1 ...
Motor_Auto_99.AnalogAlarm1.LoLo Test-993 11/9/20... SubArea1 AnalogAlarm1 D...
Motor_Auto_99.AnalogAlarm1.Lo SubArea1 AnalogAlarm1 D...
Motor_Auto_99.AnalogAlarm1.HiHi Test-988 11/9/20... SubArea1 AnalogAlarm1 D...
Motor_Auto_99.AnalogAlarm1.Hi Test-988 11/9/20... SubArea1 AnalogAlarm1 D...
Displaying 1 to 6 of 170 alarms
Default
100% Complete
Helsinki, Kyiv, Riga, Sofia, Tallinn, Vilnius

Redundant Alarm Gateway configuration setup sample

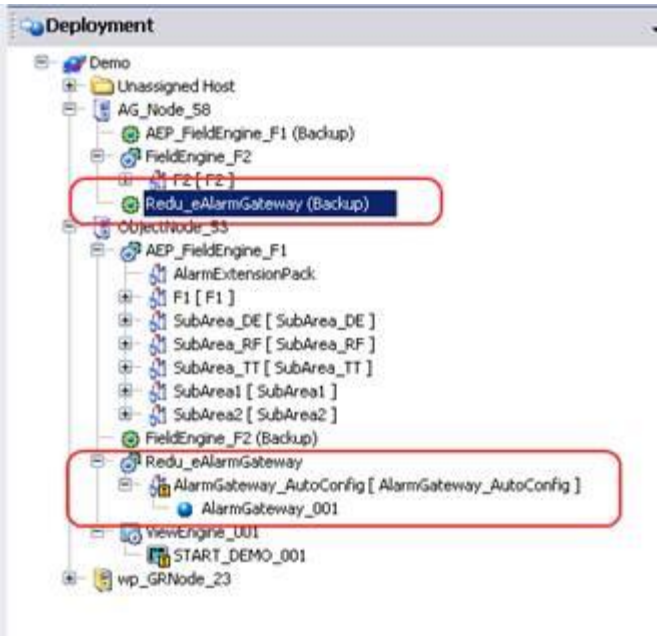
Alarm Gateway object supports WAS redundancy (for additional information please refer to WAS documentation).

1) Set up redundant AppEngines:

   a. Primary – Redu_eAlarmGateway
   b. Secondary (backup) – place on redundant partner Server.

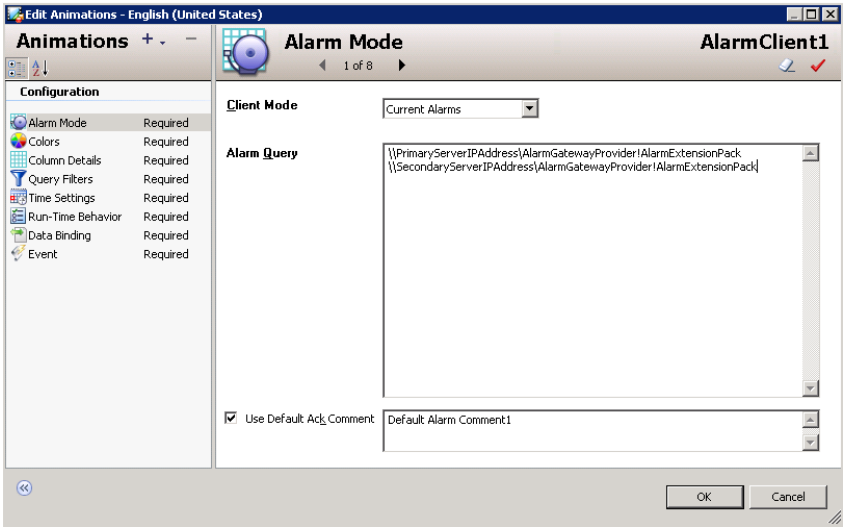


2) Configure Alarm gateway – there are on needed to configure additional settings for redundant Alarm Gateway setup:



3) Configure alarm query for alarm clients to point to Alarm Gateway on both redundant servers:

\\PrimaryServerIPAddress\AlarmGatewayProvider!AlarmExtensionPack
SecondaryServerIPAddress\AlarmGatewayProvider!AlarmExtensionPack

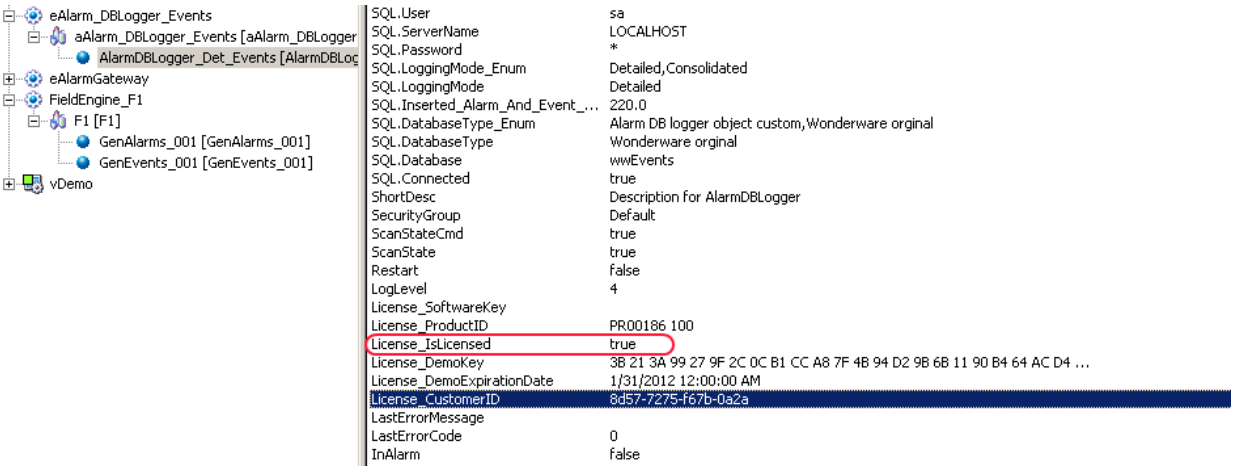


Troubleshooting

Here are common issues that may occur while using Alarm gateway object and solutions.

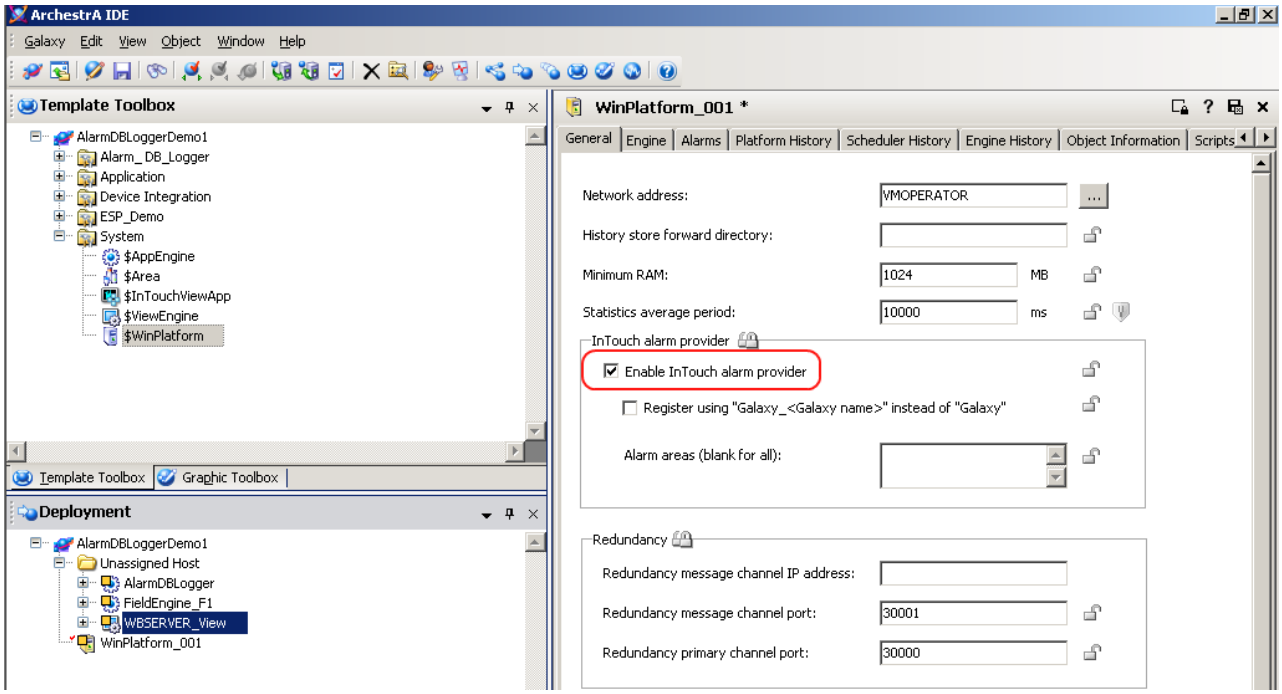
1) No alarms from Alarm Provider (WinPlatform) for Alarm Gateway.

   a) check is valid Demo or full license installed for Alarm Gateway is Attribute (**License_IsLicensed = true**) in Object Viewer:



See section **Licensing requirements** for details about object licensing.

   b) check if WinPlatform object has enabled alarming - option **Enable InTouch alarm provider** is checked.



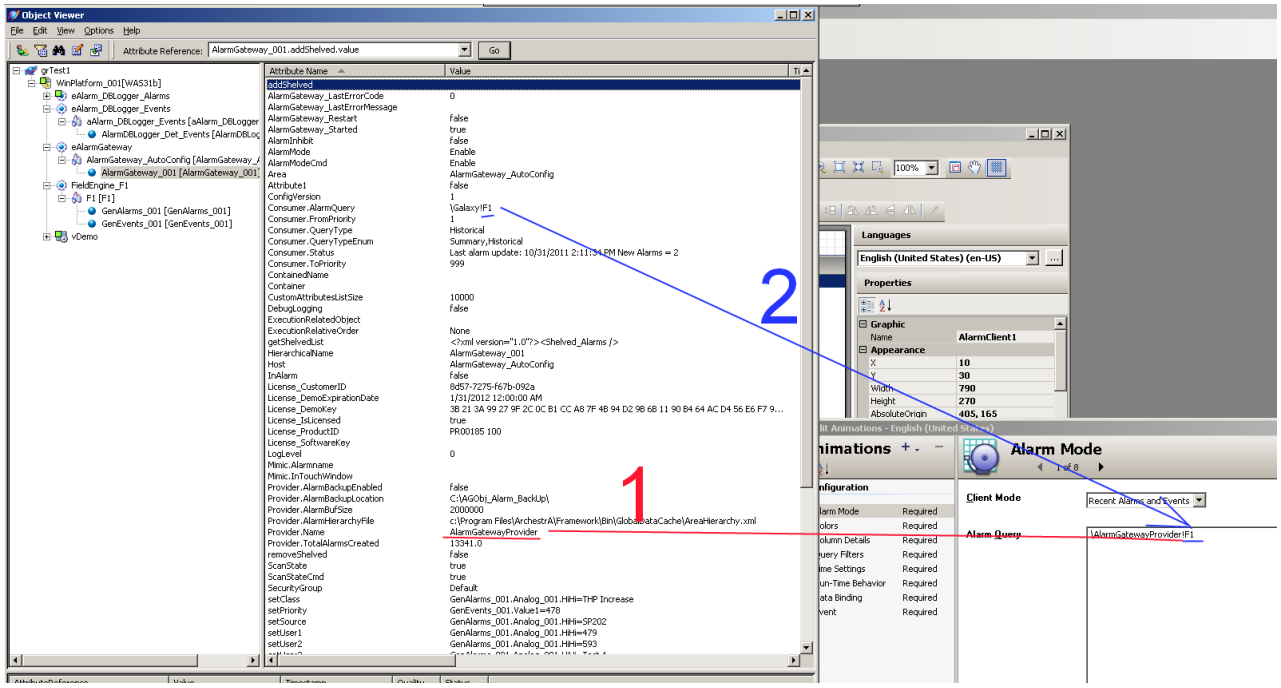
   c) if alarms are checked and alarms are displayed in any Wonderware alarm display from Platform directly, check Alarm Gateway consumer settings in Object Viewer,

is correct **alarm query**, **FromPriority, ToPriority** set (must be the same as in Wonderware alarm controls):

Consumer.ToPriority	999
Consumer.Status	Last alarm update: 10/31/2011 2:11:34 PM New Alarms = 2
Consumer.QueryTypeEnum	Summary,Historical
Consumer.QueryType	Historical
Consumer.FromPriority	1
Consumer.AlarmQuery	\Galaxy!F1
ConfigVersion	1

2) Alarms are coming to Alarm Gateway, but no alarms displayed in alarm controls that are connected to alarm gateway

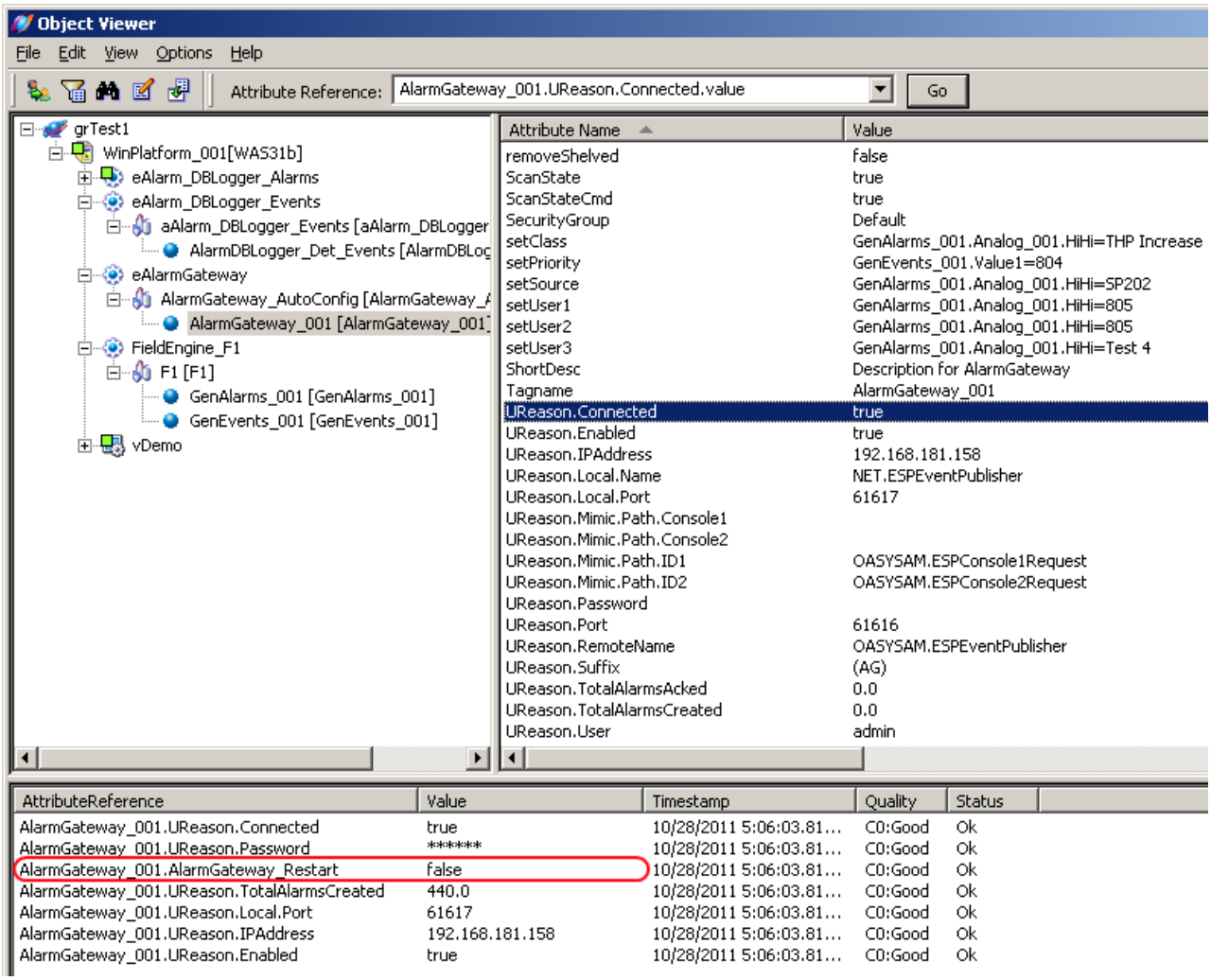
a) check are Wonderware alarm controls configured properly for Alarm gateway:
Is Alarm provider name configured properly (1 red in picture below) in Alarm control (**AlarmGatewayProvider**)
Is Alarm Areas configured properly in Alarm control must be the same as in property Consumer.AlarmAuery **F1** (2 blue in picture below).



3) Alarms are displayed into Wonderware alarm controls, but no alarms in UReason alarms system.

   a) check is UReason alarms system configured properly in Alarm Gateway – check:
      - **UReason.IPAddress** attribute **–** is correct UReason server ID address entered.
      - **UReason.Local.Port a**ttribute **–** is correct port for UReason server entered.
      - **UReason.User** attribute **–** is correct UReason user entered.
      - **UReason.password** attribute **–** is correct UReason password entered.

Note: if UReason seeting are changed is needed to restart Alarm Gateway by setting **AlarmGateway_Restart** attribute to **true**.

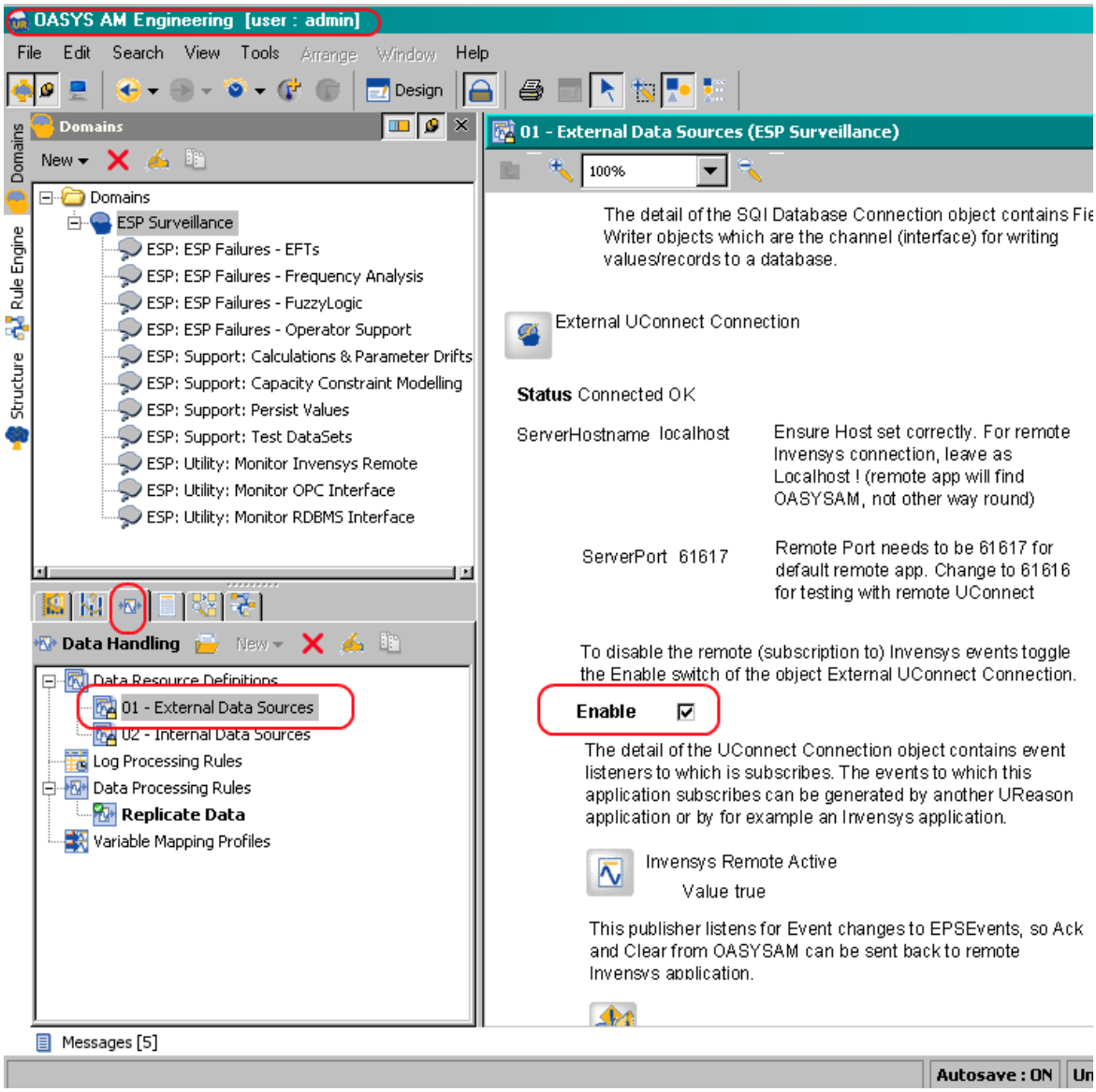


   b) If alarms are created for UReason in Alarm gateway side = **UReason.Connected = true** attribute and **UReason.TotalAlarmsCreated > 0** attribute.

and still no alarms in UReason alarming in system.

AttributeReference	Value	Timestamp	Quality	Status
AlarmGateway_001.UReason.Connected	true	10/28/2011 5:06:03.81...	C0:Good	Ok
AlarmGateway_001.UReason.Password	******	10/28/2011 5:06:03.81...	C0:Good	Ok
AlarmGateway_001.AlarmGateway_Restart	false	10/28/2011 5:06:03.81...	C0:Good	Ok
AlarmGateway_001.UReason.TotalAlarmsCreated	440.0	10/28/2011 5:06:03.81...	C0:Good	Ok
AlarmGateway_001.UReason.Local.Port	61617	10/28/2011 5:06:03.81...	C0:Good	Ok
AlarmGateway_001.UReason.IPAddress	192.168.181.158	10/28/2011 5:06:03.81...	C0:Good	Ok
AlarmGateway_001.UReason.Enabled	true	10/28/2011 5:06:03.81...	C0:Good	Ok

Is reconnected to restart connection from Wonderware alarm system from UReason alarms side in UReason console go to tab **Data handling** -> **External Data Sources** and uncheck Enable wait for ~3 sec and check again.



Advanced Troubleshooting

For advanced troubleshooting there are possible to set Log Flags - following log flags are available:

1. General – (attribute LogFlag.General) logs general logic diagnostic messages to SMC Logger.

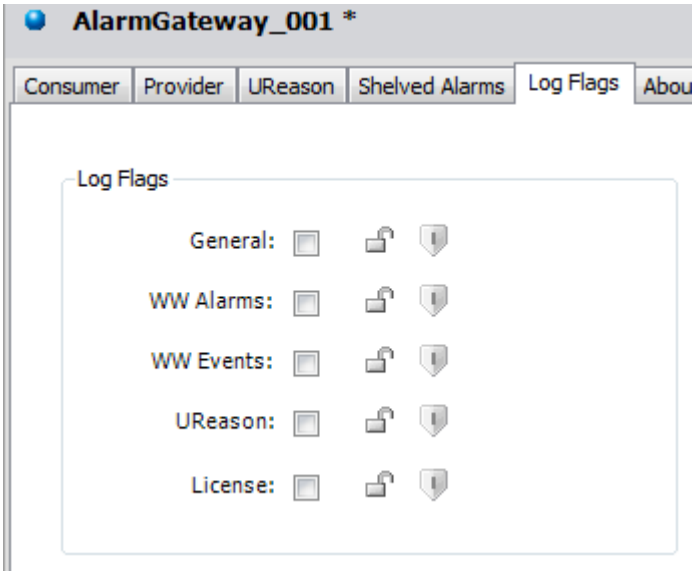
2. WW Alarms – (attribute LogFlag.WWAlarms) logs Wonderware alarms logic diagnostic messages to SMC Logger.

3. WW Events – (attribute LogFlag.WWEvents) logs Wonderware event logic diagnostic messages to SMC Logger.

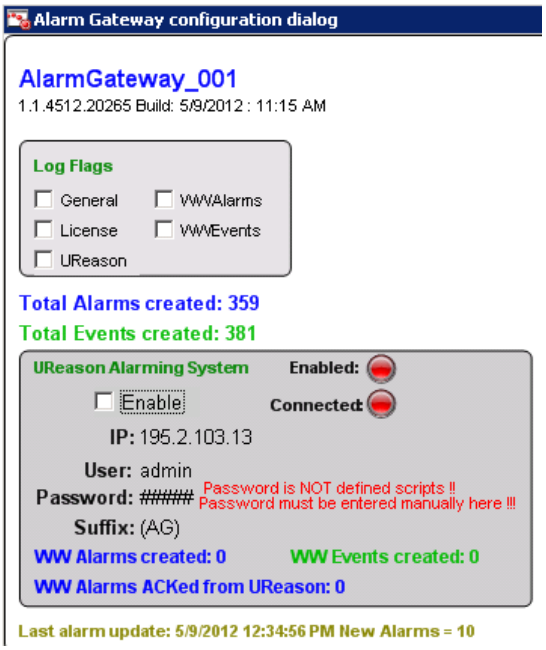
4. UReason - (attribute LogFlags.UReason) logs UReason logic diagnostic messages to SMC Logger.

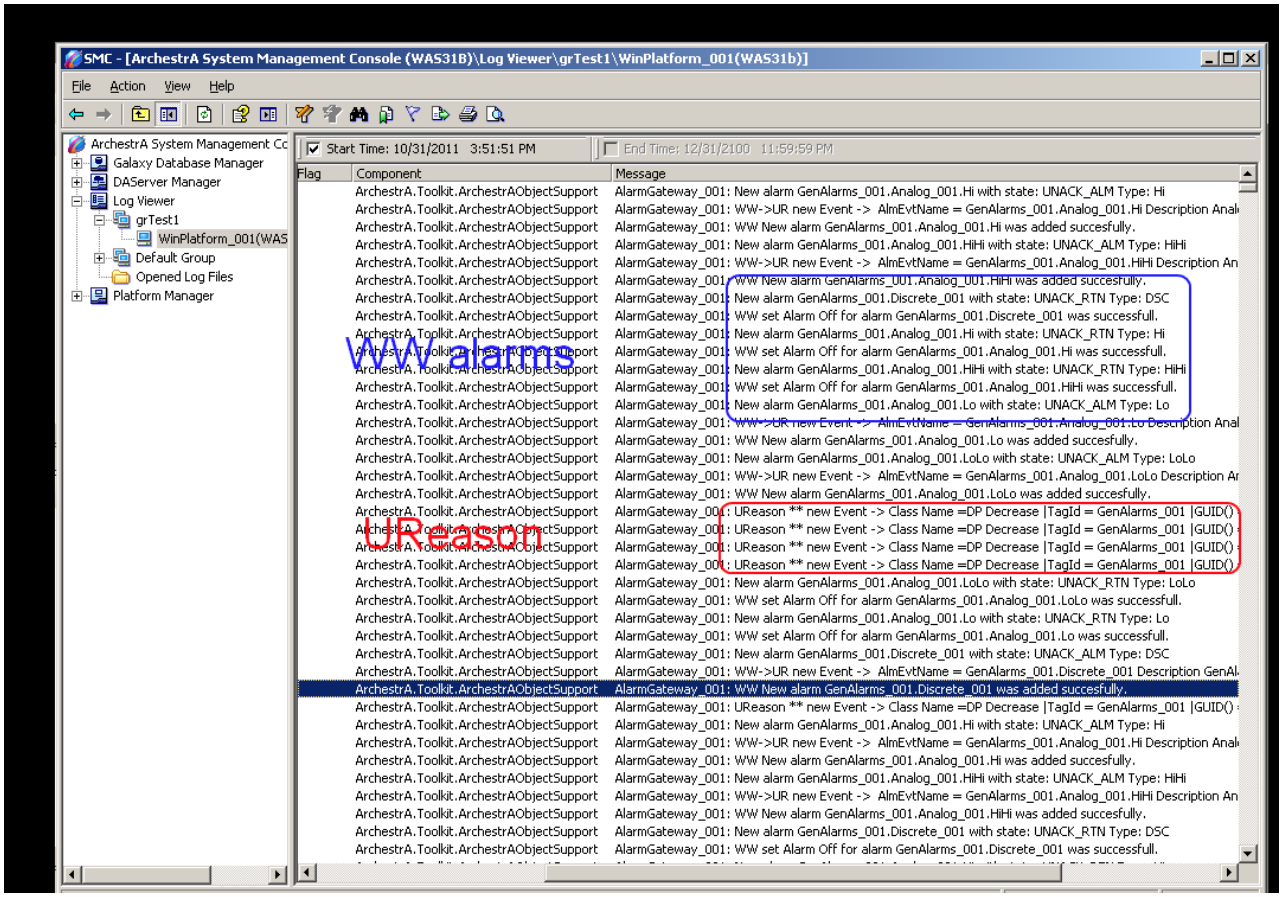
5. License - (attribute LogFlags.License) logs licensing logic diagnostic messages to SMC Logger.

Log flags are possible to configure in object editor:



Log flags is possible also configure in runtime - see ArchestrA symbol **Alarm_Gateway_Details** from Klinkmann software demo:

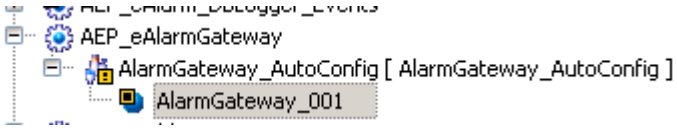

Alarm Gateway configuration dialog
AlarmGateway_001
1.1.4512.20265 Build: 5/9/2012 : 11:15 AM
Log Flags
General
WWAlarms
License
WWEvents
UReason
Total Alarms created: 359
Total Events created: 381
UReason Alarming System
Enabled:
Enable
Connected:
IP: 195.2.103.13
User: admin
Password: ###### Password is NOT defined scripts !! Password must be entered manually here !!!
Suffix: (AG)
WW Alarms created: 0
WW Events created: 0
WW Alarms ACKed from UReason: 0
Last alarm update: 5/9/2012 12:34:56 PM New Alarms = 10


SMC - [ArchestrA System Management Console (WA531B)\Log Viewer\grTest1\WinPlatform_001(WA531b)]
File
Action
View
Help
ArchestrA System Management Co
Galaxy Database Manager
DAServer Manager
Log Viewer
grTest1
WinPlatform_001(WAS
Default Group
Opened Log Files
Platform Manager
Start Time: 10/31/2011 3:51:51 PM
End Time: 12/31/2100 11:59:59 PM
Flag
Component
Message
WW alarms
UReason

Object upgrade procedure

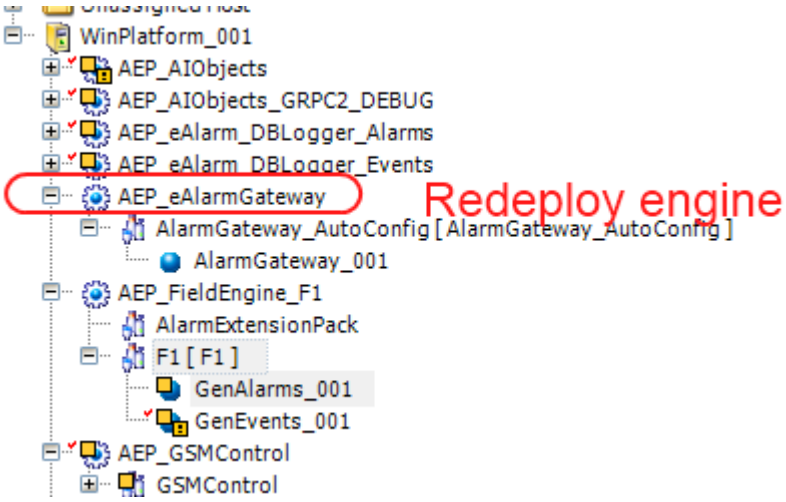
1) Open ArchestrA IDE and import new version of Alarm Gateway object.

2) Object state is changed to “Require software update”.



3) Deploy object host platform to install software update.

4) Redeploy all Application Engine(s) that are hosting Alarm Gateway object(s) to load latest AG runtime libraries are loaded and used.



Object clean uninstall procedure

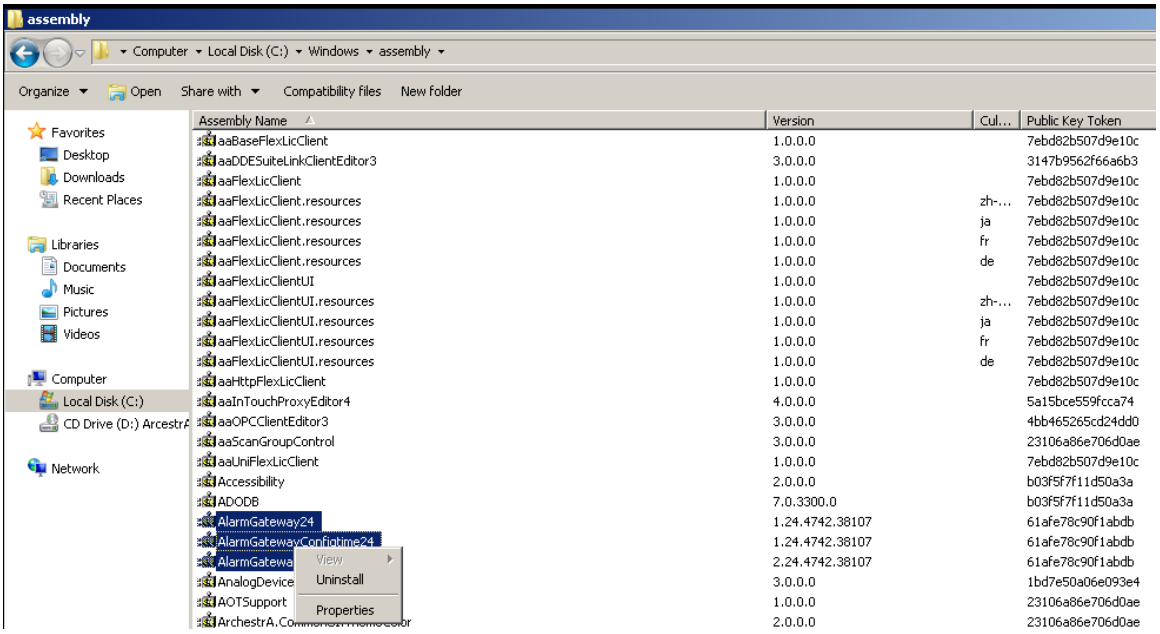
This procedure can be used to fully uninstall the object.
Can be used to downgrade to previous object version or to fix object upgrade issues.

Note: must be performed on **all WAS nodes where object was used/installed** (IDE node, object nodes).

1) Undeploy all **WinPlatforms** that host AG object.
2) Delete all instances and templates from Galaxy.
3) Close ArchestrA IDE.
4) Uninstall manually AlarmGateway object assembles in case if they still are installed:

   a. Browse to folder **C:\Windows\assembly**, select all Assemblies named **AlarmGatewayXX** were XX is object version number e.g. 24.

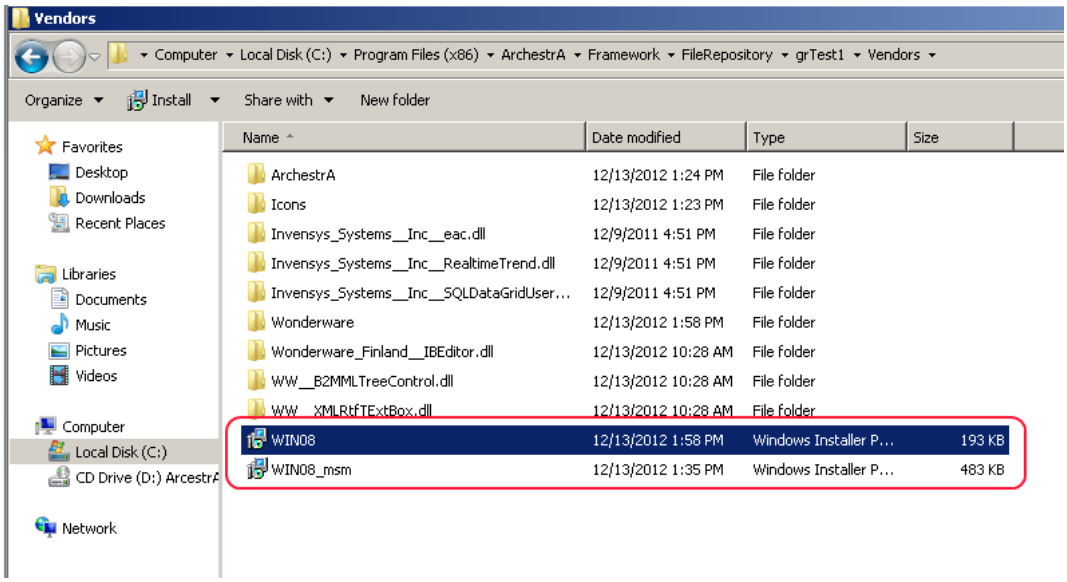
   b. Open context menu by clicking right mouse button and select option **Uninstall.**



5) Delete WAS generated MSI installers that contains links to deleted/invalid files from: **C:\Program Files (x86)\ArchestrA\Framework\FileRepository\<Your Galaxy Name>\Vendors**

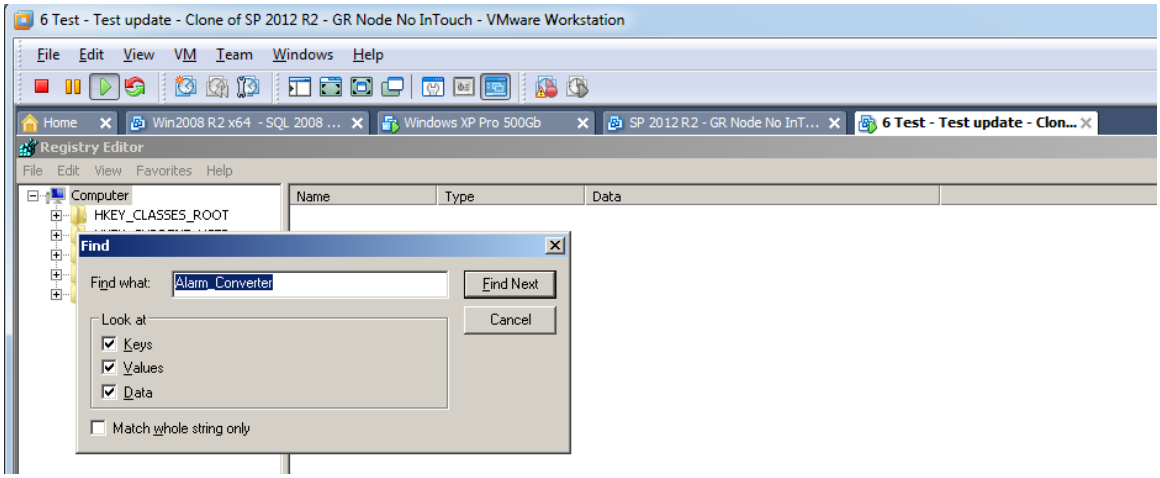
Where is **<Your Galaxy Name>** your galaxy name e.g. **grTest1**

<User name> - currently logged user name – user that was used to install object.



6) Delete temp files from C:\Windows\Temp

7) Delete all entries that contains **Alarm_Converter** from Windows registry.



8) Restart PC.

WONDERWARE FINLAND
Alarm Gateway Object
Revision History

Jun 2011	Rev 1.0	First Release
Jun 2011	Rev 1.1	Alarm Group Hierarchy XML “Associated Attribute” and “Description” changed
Sep 2011	Rev 1.2	“Custom attributes” and “UReason gateway” added.
Sep 2011	Rev 1.3	“Custom attributes” for “UReason gateway” added. Mimic windows functionality added.
Oct 2011	Rev 1.4	“Installing the Alarm Gateway Object” section modified. “Troubleshooting” and “Object upgrade procedure” sections added.
Mar 2012	Rev 1.5	Exposed Alarm field feature added.
Jul 2012	Rev 1.6	Updated Exposed attributes section.
Aug 2012	Rev. 1.7	added section Custom Acked alarm comment field.
Oct 2012	Rev. 1.9	Object version 17 updates.
Dev 2012	Rev. 1.10	Added object clean uninstall procedure.